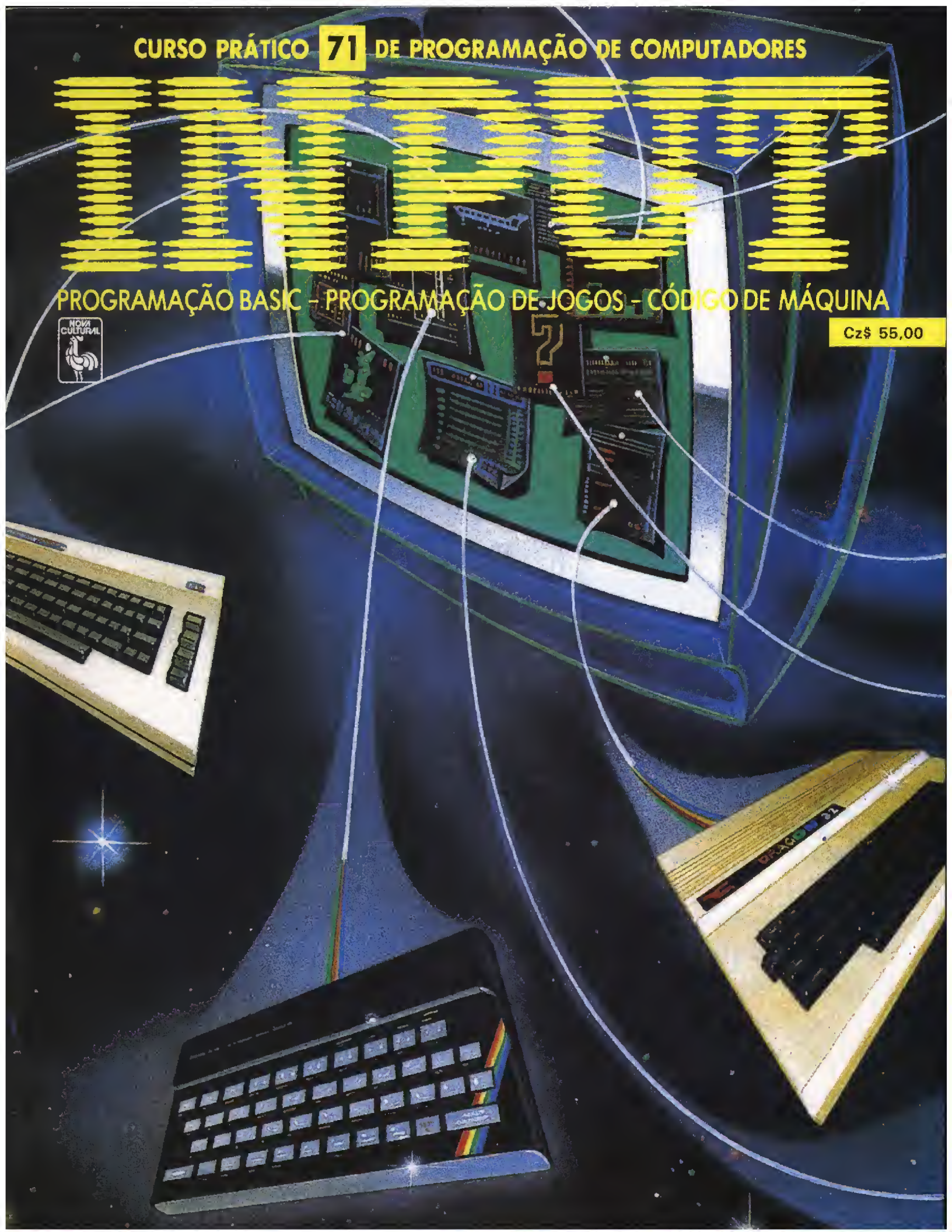


CURSO PRÁTICO 71 DE PROGRAMAÇÃO DE COMPUTADORES

PROGRAMAÇÃO BASIC - PROGRAMAÇÃO DE JOGOS - CÓDIGO DE MÁQUINA

Cz\$ 55,00



INPUT

Vol. 5

Nº 71

NESTE NÚMERO

PROGRAMAÇÃO BASIC

MAIS OPERAÇÕES COM CADEIAS

READ e DATA. Menus e opções. Substituição de subcadeias 1401

SOFTWARE

TROCA DE MENSAGENS

Quadro de avisos. Modem. Menus..... 1404

APLICAÇÕES

UM EDITOR MUSICAL (2)

Segunda parte do programa..... 1408

PROGRAMAÇÃO BASIC

OS SEGREDOS DO TRS-80 (5)

Desativação do <BREAK>. Auto-Repetição... 1412

PROGRAMAÇÃO DE JOGOS

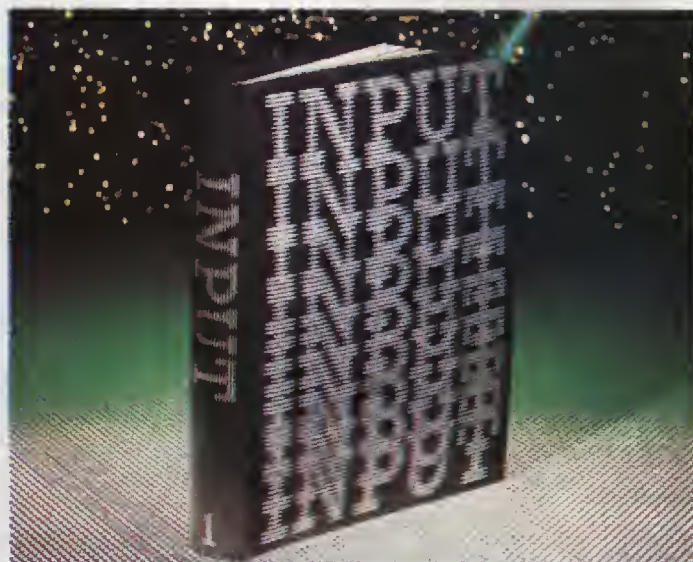
COMPRESSÃO DE TEXTOS (2)

Frequência de pares. Método chinês..... 1414

PROGRAMAÇÃO BASIC

ROTINAS EM CÓDIGO DE MÁQUINA (2)

Rotinas nas linhas **DATA. PEEK e POKE**..... 1419



PLANO DA OBRA

INPUT é uma obra editada em fascículos semanais, e cada conjunto de 15 fascículos compõe um volume. A capa para encadernação de cada volume estará à venda oportunamente.

FÉRIAS, VIAGENS, MUDANÇAS...

NÃO FIQUE COM A COLEÇÃO INCOMPLETA

Se você está saindo de férias, pretende viajar ou vai se ausentar por algum tempo, avise antecipadamente seu jornaleiro. Ele pode guardar os seus fascículos enquanto você estiver fora. Se, por qualquer motivo, você perdeu alguns números, peça-os também a seu jornaleiro, ou entre em contato com nossa Distribuidora:

1. **Pessoalmente** — Em **São Paulo**, os endereços são: rua Brigadeiro Tobias, 773, Centro; av. Industrial, 117, Santo André. No **Rio de Janeiro**, av. Mem de Sá, 191/193, Centro.
2. **Por carta** — Envie para:
DINAP — Distribuidora Nacional de Publicações
Números Atrasados
Estrada Velha de Osasco, 132 — Jardim Teresa
CEP 06040 — Osasco — SP
3. **Por telex** — Utilize o nº (011) 33 670 DNAP.

Em **Portugal**, os pedidos devem ser feitos à Distribuidora Jardim de Publicações Lda. — Qta. Pau Varais, Azinhaga de Fetais, 2685, Camarate, Lisboa; Apartado 57; Telex 43 069 JARLIS P.

Atenção: Após seis meses do encerramento da coleção, o atendimento dos pedidos dependerá da disponibilidade do estoque.

Obs.: Quando pedir livros, mencione sempre o título e/ou autor da obra, além do número da edição.

COLABORE CONOSCO

Encaminhe seus comentários, críticas, sugestões ou reclamações ao

SERVIÇO DE ATENDIMENTO AO LEITOR

Caixa Postal 9 442, São Paulo — SP. CEP 01051.



EDITOR
RICHARD CIVITA

NOVA CULTURAL

Presidente

Flávio Barros Pinto

Diretoria

Carmo Chagas, Iara Rodrigues,
Pierluigi Bracco, Plácido Nicoletto,
Roberto Silveira, Shoji Ikeda,
Sônia Carvalho

REDAÇÃO

Diretor Editorial: Carmo Chagas

Editores Executivos:

Antonio José Filho, Berta Sztark Amar

Editor Chefe: Paulo de Almeida

Editoras Assistentes: Ana Lúcia B. de Lucena,

Marisa Soares de Andrade

Chefe de Arte: Carlos Luiz Batista

Assistentes de Arte: Dagmar Bastos Sampaio,
Grace Alonso Arruda, Monica Lenardon Corradi

Colaboradores

Consultor Editorial Responsável:

Dr. Renato M. E. Sabbatini

(Diretor do Núcleo de Informática Biomédica da
Universidade Estadual de Campinas)

Execução Editorial: DATAQUEST Assessoria
em Informática Ltda., Campinas, SP

Tradução, adaptação, programação e redação:

Abílio Pedro Neto, Aluísio J. Dornellas de Barros,
Marcelo R. Pires Therezo, Marcos Huascar Velasco,
Raul Neder Porrelli, Ricardo J. P. de Aquino Pereira

Coordenação Geral: Rejane Felizatti Sabbatini

COMERCIAL

Diretor Comercial: Roberto Silveira

Gerente Comercial: Flávio Maculan

Gerente de Circulação: Denise Mozol

Gerente de Propaganda e Publicidade: José Carlos Madio

Gerente de Pesquisa e Análise de Mercado:

Wagner M. P. Nabuco de Araújo

CLC

A Editora Nova Cultural Ltda. é uma empresa do
Grupo CLC — Comunicações, Lazer e Cultura

Presidente: Richard Civita

Diretoria: Flávio Barros Pinto, João Gomez,
Menahem M. Politi, Renê C. X. Santos,
Stéfio Alves Campos

© Marshall Cavendish Limited 1984/85.

© Editora Nova Cultural Ltda., São Paulo, Brasil, 1986.

Edição organizada pela Editora Nova Cultural Ltda.

Av. Brigadeiro Faria Lima, nº 2000 - 3º andar

CEP 01452 - São Paulo - SP - Brasil

(Artigo 15 da Lei 5 988, de 14/12/1973).

Esta obra foi composta na AM Produções Gráficas Ltda.
é impressa na Divisão Gráfica da Editora Abril S.A.

MAIS OPERAÇÕES COM CADEIAS

Apresentamos neste artigo algumas funções e rotinas relativamente simples, que podem facilitar bastante as tarefas que envolvem a manipulação de cadeias de caracteres. Conheça-as.

A maneira mais adequada de armazenar conjuntos de dados numéricos ou alfanuméricos dentro de um programa em BASIC consiste em colocá-los em linhas DATA, e lê-los com comandos READ.

Porém, quando o número de dados é pequeno, não compensa introduzi-los em linhas DATA e acrescentar ao programa comandos destinados a lê-los e carregá-los em variáveis indexadas. É muito mais conveniente colocar os dados dentro de uma única variável string, de tal forma que possamos recuperá-los ou consultá-los por meio de uma sub-rotina ou função.

Esta é também a melhor solução para os usuários de computadores que não dispõem dos comandos READ e DATA, como os micros pessoais ZX-81.

TRINTA DIAS TEM SETEMBRO...

O programa que se segue exemplifica bem o procedimento sugerido. Ele informa quantos dias tem um determinado mês do ano:

SS

```
20 LET CS="312931303130313130313031"
30 PRINT "ENTRE O NUMERO DO MES ";
45 INPUT M
50 IF M<1 OR M>12 THEN GOTO 40
60 PRINT "ESTE MES TEM ";
70 PRINT CS((M-1)*2+1 TO (M-1)*2+2); " DIAS."
80 GOTO 40
```

TTT

```
20 CS="312931303130313130313031"
40 INPUT "ENTRE O NUMERO DO MES
```

";M

```
50 IF M<1 OR M>12 THEN GOTO 40
60 PRINT "ESTE MES TEM ";
70 PRINT MIDS(CS, (M-1)*2+1, 2);
"DIAS."
80 GOTO 40
```

Os números de dias referentes a cada mês são colocados na cadeia de caracteres CS. A função da linha 70 extrai o número de dias do mês M (especificado no programa de teste, pela entrada da linha 40) usando a expressão:

posição inicial em MS = (M - 1) * 2 + 1
número de dígitos = 2

Note que, se M = 1, a posição inicial em MS é 1. Esta é uma fórmula geral, que pode ser usada para extrair subcadeias de comprimento fixo, concatenadas em uma única cadeia.

Nos computadores que têm funções programáveis do tipo string, essa fórmula pode ser colocada dentro de uma função, e usada repetidas vezes, como neste programa:

TTT

```
10 DEF FNT$(I,L,SS)=MIDS(SS,(I-1)*L+1,L)
20 LET MS="JANFEVMARABRMAIJUNJULAGOSETOOUTNOVDEZ"
30 LET CS="312931303130313130313031"
40 INPUT "ENTRE O NUMERO DO MES ";M
50 IF M<1 OR M>12 THEN GOTO 80
60 PRINT FNT$(M,3,MS); " = "; FNT$(M,2,CS); " DIAS."
70 GOTO 40
80 PRINT:PRINT "DATA INVALIDA":GOTO 40
```

S

Para executar o programa no Spectrum, substitua a linha 10:

```
10 DEF FNT$(I,L,SS)=SS((I-1)*L+1 TO (I-1)*L+L)
```

Na função FNT\$, definida na linha 10, o primeiro argumento é o índice; o segundo, o tamanho fixo das subcadeias e o terceiro, o nome da cadeia com os dados. Note que a mesma função é usa-

■	READ E DATA
■	MENUS E OPÇÕES
■	RECUPERAÇÃO DE PALAVRAS
■	O USO DE MACROS
■	A INSTRUÇÃO CLEAR

da com duas cadeias de formatos diferentes, na linha 60 do programa.

Em computadores que não aceitam funções com mais de um argumento, pode-se empregar uma sub-rotina, mais longa, mas igualmente útil:

TTT

```
20 LET MS="JANFEVMARABRMAIJUNJULAGOSETOOUTNOVDEZ"
30 LET CS="312931303130313130313031"
40 INPUT "ENTRE O NUMERO DO MES ";M
50 IF M<1 OR M>12 THEN GOTO 98
55 LET I=M
60 LET L=3
65 LET XS=MS
70 GOSUB 100
75 PRINT SS;" TEM ";
80 LET XS=CS
85 GOSUB 100
90 PRINT SS;" DIAS."
95 GOTO 40
98 PRINT:PRINT "DATA INVALIDA"
99 GOTO 40
100 LET SS=MIDS(XS,(I-1)*L+1,L)
110 RETURN
```

SS

Para rodar o programa nos micros Spectrum e ZX-81, substitua a linha 100 do programa:

```
100 LET SS=XS((I-1)*L+1 TO (I-1)*L+L)
```

MENUS E OPÇÕES

Outra aplicação bastante interessante dos conceitos aqui discutidos consiste na programação rápida de menus com entradas por extenso. As várias opções, colocadas em uma única variável string, são depois recuperadas pelo método já descrito:

TTT

```
10 LET FLS="INSEREEDITARAPAGARLISTAR"
20 PRINT "ENTRE UM COMANDO: ";F
```



```

LS: " ":
30 INPUT OPS
40 FOR I=1 TO 4
50 LET DS=MID$(FL$, (I-1)*6+1,6)
60 IF DS=OPS THEN GOTO 80
65 NEXT I
70 PRINT "CODIGO INVALIDO"
75 GOTO 20
80 PRINT "COMANDO ESCOLHIDO =>";DS
90 GOTO 20

```



Substitua a seguinte linha do programa para micros da linha Sinclair:

```
50 LET DS=FL$((I-1)*6+1 TO (I-1)*6+6)
```

Para os micros que dispõem da instrução **INSTR**, que localiza subcadeias, e de funções programáveis, podemos fazer um programa mais compacto:



```

5 DEF FNC(OS,CS,N)=INT((INSTR(OS,LEFT$(CS+STRING$(N," ").N))-1)/N)+1
10 OS="CR CH PD DB AT "
20 PRINT "ENTRE O CODIGO DA TRANSAÇÃO ";
30 PRINT "(CODIGOS VALIDOS: ";OS;")";
40 LINE INPUT "CODIGO : ";CS
50 I=FNC(OS,CS,3)
60 IF I=0 THEN PRINT "*** CODIGO INVALIDO":GOTO 30
70 PRINT "CODIGO ";I
80 GOTO 20

```

A função **FNC\$** retorna o número da opção (posição sequencial na cadeia). Seus argumentos de entrada são:

OS: cadeia que contém as opções;
CS: subcadeia procurada em **OS**;
N: tamanho fixo das subcadeias.

É necessário colocar espaços em branco para separar os comandos armazenados em **OS**. Se isso não for feito, a função **INSTR** pode encontrar subcadeias superpostas, que não correspondem às opções possíveis.

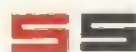
RECUPERAÇÃO DE PALAVRAS

A recuperação de palavras, códigos ou números de uma lista, efetuada de acordo com o método que acabamos de explicar, requer que todos eles tenham o mesmo comprimento, de modo a permitir a aplicação de uma fórmula.

Isso nem sempre é desejável ou pos-

sível. Suponhamos que, na programação de um jogo de aventura, você peça ao jogador que entre a lista dos objetos que ele quer levar em uma incursão a um castelo mal-assombrado. Nesse caso, os comprimentos das palavras contidas na cadeia de entrada podem ser bastante diferentes entre si, o que exigiria a elaboração de um programa um pouco mais "inteligente".

Se determinarmos, por exemplo, que o caractere de separação entre palavras é o espaço em branco, a seguinte rotina realizará o trabalho:



```

80 REM ----- SUBROTINA -----
100 LET A$=""
110 IF CSS(1)="" THEN RETURN
120 IF CSS(1)<>" " THEN GOTO 130
125 LET CSS=CSS(2 TO)
127 GOTO 100
130 LET A$=A$+CSS(1)
135 LET CSS=CSS(2 TO)
140 IF CSS(1)="" OR CSS(1)=" " THEN RETURN
150 GOTO 130

```



```

80 REM ----- SUBROTINA -----
100 A$=""
110 IF MID$(CSS,1,1)="" THEN RETURN
120 IF MID$(CSS,1,1)=" " THEN CSS=MID$(CSS,2):GOTO 100
130 A$=A$+MID$(CSS,1,1):CSS=MID$(CSS,2)
140 IF MID$(CSS,1,1)="" OR MID$(CSS,1,1)=" " THEN RETURN
150 GOTO 130

```

A cadeia de entrada é **CSS**. A cada chamada da rotina, **A\$** retorna uma subcadeia contida entre dois espaços em branco, em **CSS**; extraíndo-a, reduz o tamanho de **CSS**.

Eis um programa que testa o funcionamento dessa rotina:



```

20 PRINT "ENTRE TRES COISAS QUE CARREGARA (SEPARADAS POR ESPACOS) : "
30 INPUT CSS
40 LET N=0
50 PRINT"VOCE CARREGARA: "
55 GOSUB 100
60 IF A$="" THEN GOTO 80
70 LET N=N+1:PRINT N,A$
75 GOTO 60
80 PRINT "E MAIS NADA..."
90 STOP

```



Para rodar nesses computadores, acrescenta a seguinte linha:

```
10 CLEAR 1000
```

A linha 100 inicializa a cadeia de saída, **A\$**. A linha 110 começa um laço de repetição, testando se a cadeia de entrada, **CSS**, é vazia — ou seja, se todas as palavras nela contidas já foram extraídas. Em caso afirmativo, a sub-rotina retorna o controle ao programa que a chamou. A linha 120 verifica se o primeiro dos caracteres que restaram em **CSS** é um espaço em branco. Se for, ele é "podado" de **CSS**, e o programa retorna para a linha 100, para executar um novo teste. Se o primeiro caractere não for um espaço em branco, a linha 130 retira-o de **CSS**, levando-o para **A\$**. Finalmente, se um novo espaço em branco for encontrado (fim de subcadeia), ou se **CSS** tiver se esgotado, a linha 140 retorna o controle ao seu programa principal.

SUBSTITUIÇÃO DE SUBCADEIAS

Uma das funções mais úteis de um programa de processamento de textos é a que permite substituir todas as ocorrências de uma determinada palavra, ou sequência de caracteres, em um documento. Essa função possibilita, entre outras coisas, o uso de *macros* para a redação rápida de textos muito repetitivos. Digamos que você esteja escrevendo um artigo sobre processamento de imagens, e que essa expressão ("processamento de imagens") será citada algumas dezenas de vezes, por extenso, ao longo do texto. Para poupar tempo e esforço, sempre que for preciso digitá-la, você poderá colocar um símbolo qualquer em seu lugar — por exemplo, dois caracteres que dificilmente aparecerão juntos em outras partes do texto: ***P**, **\$** etc. Depois, bastará acionar a função de busca e substituição, e o programa trocará o símbolo convencionado pela expressão por extenso, sempre que o encontrar.

Programar essa operação em BASIC não chega a ser difícil, mas envolve algumas complicações, pois a sequência que irá substituir o símbolo no texto pode ter um número de caracteres menor, igual ou maior. Além disso, a maioria dos microcomputadores não aceita variáveis string, com mais de 255 caracteres. Assim, durante o processo de substituição, é necessário fazer com que o programa verifique esse número, para

não ocorrer nenhum erro. Apresentamos, a seguir, uma sub-rotina de busca e substituição, que você poderá adicionar a seus programas:

SS

```
100 LET N=1
110 IF LEN A$ -LEN B$ +LEN C$ >
255 THEN RETURN
120 FOR I=N TO LEN A$
130 IF A$(I TO I+LEN B$-1)=B$ T
HEN GOTO 160
140 NEXT I
150 RETURN
160 LET A$=A$(TO I-1)+C$+A$(I+L
EN B$ TO)
170 LET N=I+LEN C$
180 GOTO 110
```

TT

```
100 N=1
110 IF LEN(A$)-LEN(B$)+LEN(C$)>
255 THEN RETURN
120 FOR I=N TO LEN(A$)
130 IF MIDS(A$,I,LEN(B$))-B$ TH
EN GOTO 160
140 NEXT I:RETURN
160 A$=LEFT$(A$,I-1)+C$+MIDS(A$
,I+LEN(B$))
170 N=I+LEN(C$)
180 GOTO 110
```

Os argumentos de entrada da sub-rotina são:

A\$ - cadeia que terá substituições;
B\$ - subcadeia em **A\$** a substituir;
C\$ - subcadeia que substituirá **B\$**.

Suponhamos que a célebre frase de Gertrude Stein:

UMA ROSA E' UMA ROSA E' UMA ROSA

deverá ser transformada em outra (não menos poética):

UMA PEDRA E' UMA PEDRA E' UMA PEDRA

Neste exemplo, então:

A\$ = UMA ROSA E' UMA ROSA E' UMA ROSA
B\$ = ROSA
C\$ = PEDRA

A rotina funciona da seguinte maneira: a linha 100 inicializa **N** em 1. Essa variável aponta o caractere de **A\$** a partir do qual **B\$** será procurado. Na linha 110, checamos se o comprimento total da cadeia substituída não excede 255. O laço que vai da linha 120 à 140 verifica se **B\$** está presente em **A\$**, a partir do caractere **N** até o fim. Se estiver, a li-

nha 160 efetua a substituição, concatenando a primeira seção de **A\$**, até a ocorrência de **B\$**, mais **C\$** e a parte restante de **A\$**. **N** é então atualizado e ocorre nova busca por **B\$** (retorno à linha 110).

TT

Nos micros com função **INSTR** (*instring*), o laço das linhas 120 a 140 pode ser substituído por uma só linha:

```
120 I=INSTR(I,A$,B$):IF I=0 THE
N RETURN
```

Esta listagem testa a sub-rotina de substituição. Ela é auto-explicativa:

SS

```
20 PRINT "ENTRE UMA FRASE ";
25 INPUT A$
30 PRINT "PALAVRA PARA SUBSTITUIR ";
35 INPUT B$
40 PRINT "POR ";
45 INPUT C$
50 GOSUB 100
60 PRINT A$
70 GOTO 20
```

TT

Para rodar o programa no TRS-80 e no TRS-Color, adicione esta linha:

```
10 CLEAR 1000
```

Se você quiser substituir todas as ocorrências de uma subcadeia, em um texto composto de várias linhas (armazenadas em **DATA**, por exemplo), utilize o próximo programa. Não se esqueça de colocar uma última linha **DATA ""** para assinalar o fim do texto:

ST

```
10 INPUT "PALAVRA PARA SUBSTITUIR ";
20 INPUT "POR ";
30 READ A$
40 IF A$="" THEN GOTO 70
50 GOSUB 70:PRINT A$
60 GOTO 30
70 STOP
```

A INSTRUÇÃO CLEAR

A programação de variáveis string apresenta algumas diferenças conforme o tipo de interpretador BASIC usado.

Como vimos no artigo anterior desta série (página 1214), os interpretadores se dividem em dois grandes grupos

quanto às funções de tratamento de variáveis literais. No primeiro grupo, chamado Microsoft (nome da *software house* norte-americana responsável pelo desenvolvimento do BASIC padrão para os micros MSX, TK-2000, Apple II, TRS-80 e TRS-Color, entre outros), encontramos funções familiares como **LEFT\$, RIGHT\$, MID\$, INSTR** etc.

No segundo grupo, denominado Sinclair (destinado às máquinas compatíveis com as linhas ZX-81 e Spectrum), a cadeia de caracteres é tratada como um conjunto dimensionado, e a partícula **TO** se encarrega da referência às subcadeias de um string.

Os interpretadores do tipo Microsoft também diferem dos do tipo Sinclair quanto ao gerenciamento do espaço disponível na memória RAM para variáveis literais e numéricas. Ambos, porém, armazenam uma variável literal de forma semelhante (veja o artigo da página 1101): uma cadeia pode ter entre 0 e 255 caracteres, armazenados em um conjunto contíguo de bytes. O primeiro byte de um string é usado para armazenar o seu comprimento, ou seja, o número de caracteres que possui. Essa informação é usada, por exemplo, pela função **LEN**, disponível nos dois tipos de interpretador, que informa o comprimento da cadeia. Se uma variável string for concatenada através de uma expressão como **LET A\$=A\$+B\$**, o interpretador manipula **A\$** e **B\$** de modo a transferi-los para uma área livre da memória RAM, colocá-los um após o outro, eliminar o byte de comprimento do segundo string (**B\$**) e atualizar o valor armazenado no byte de comprimento do primeiro string (**A\$**).

Como se pode concluir, as operações de concatenação são lentas, devendo ser evitadas quando possível. Além disso, exigem uma área livre de memória para a execução da cópia — os originais dos dois strings **A\$** e **B\$** continuam no mesmo lugar, mas "desativados" pelo interpretador. Periodicamente, uma função chamada "coleta de lixo" (*garbage collection*, em inglês) recupera essas áreas para uso pelo programa.

Quanto à reserva de espaço para strings, na RAM, os interpretadores também apresentam diferenças. Os do tipo Sinclair operam automaticamente ou por meio de uma declaração **DIM**, que deve ser usada para dimensionar strings. Os do tipo Microsoft nem sempre precisam de uma declaração explícita para dimensionar o tamanho da área de strings. Nos micros das linhas TRS-80 e TRS-Color, a instrução **CLEAR** serve a esse propósito. No Apple e no MSX, ela não é necessária.

TROCA DE MENSAGENS

No artigo *Sua ligação com o mundo* (página 561), examinamos como o dispositivo especial chamado *modem* pode conectar um micro a outros computadores, utilizando a linha telefônica. Através dele, uma grande variedade de bancos de dados computadorizados ficará à disposição do usuário: cotações das Bolsas, noticiários, referências bibliográficas sobre assuntos técnicos etc. Existem hoje, em todo o mundo, mais de um milhão de pessoas que se beneficiam de redes de computadores.

O acesso a esses sistemas, entretanto, não é algo exatamente barato, principalmente quando é necessário pagar ligações internacionais (DDI). Além disso, muitos usuários se ressentem da falta de um contato mais pessoal entre quem fornece e quem recebe a informação — o que talvez explique, por exemplo, o sucesso e a popularidade do rádioamadorismo. Contudo, já existe um equivalente desse hobby dentro do campo das redes informatizadas de dados. Trata-se da versão "doméstica" dos grandes sistemas telemáticos (como CompuServe e Cirandão) denominada *quadro de avisos computadorizado* (do inglês, *bulletin board*).

Um *Computerized Bulletin Board Service* (ou CBBS, como é mais conhecido entre os aficionados) é, em sua forma mais simples, o equivalente eletrônico do quadro de avisos de um clube, onde todos podem afixar notícias, recados etc. Para ter acesso a um sistema desse tipo, o usuário precisa — além de um telefone e de um micro — dos seguintes elementos: um modem, um software específico para intercomunicação e o número do telefone de um CBBS. O custo do acesso propriamente dito (ler e/ou escrever no quadro de avisos), geralmente muito baixo, varia segundo a finalidade do serviço (comercial ou não) e inclui a despesa normal com a ligação telefônica.

Embora os CBBS existam há algum tempo (sobretudo nos EUA, onde foram criados), em nosso país eles ainda são uma novidade. Apenas as cidades maiores, como São Paulo e Rio de Janeiro, dispõem desse serviço. Tudo indica, entretanto, que os CBBS se multiplicarão rapidamente e que a maioria

dos proprietários de modems tomará contato com eles brevemente.

CARREGAR E DESCARREGAR

Os CBBS não são utilizados somente como sistemas de mensagens. Na realidade, sua rápida expansão se deve, principalmente, ao fato de a maioria deles dispor de um serviço especial que possibilita a transmissão de software de um computador para outro. No jargão dos seus aficionados, *uploading* significa o ato de enviar um software para armazenamento no CBBS, enquanto *downloading* quer dizer retirar uma cópia das informações disponíveis. É importante ressaltar aqui que o micro que manda e o que recebe a listagem do programa não precisam ser compatíveis entre si. Essa é, portanto, uma oportunidade rara de transportar certos tipos de software por meios incomuns.

O princípio dessa idéia consiste em converter os programas em textos ASCII (padronizados para todos os micros, que abordamos, menos o ZX-81) para realizar a transmissão via modem.

Outro requisito para o trabalho de copiar os programas disponíveis no CBBS é ter uma unidade de discos e um interpretador ou compilador da linguagem original do programa. Com eles, poderemos carregar arquivos em formato ASCII. Os sistemas operacionais das linhas TRS-80, TRS-Color e MSX oferecem essa facilidade sem requerer o emprego de programas conversores especiais. Em outros micros, o próprio software de transmissão pode incluir uma pequena rotina que transforma texto em programas e vice-versa. Além disso, alguns CBBS são específicos para uma linha de micros (os da linha Apple e IBM PC são os mais comuns), e permitem a transferência direta de programas entre dois computadores compatíveis, sem o arquivo ASCII intermediário.

COMO FUNCIONA UM CBBS

O quadro de avisos geralmente é "hospedado" por um microcomputador ligado, por um ou mais modems, a um

Compartilhe informações profissionais e de lazer com outras pessoas através de seu micro. Basta ligá-lo por telefone a um dos vários serviços computadorizados de "quadro de avisos"

tronco telefônico. Para baratear os custos operacionais, a maioria dos CBBS atende simultaneamente a um número limitado de usuários — normalmente entre 1 e 5. À primeira vista, isso parece pouco, porém, como veremos adiante, é uma impressão falsa já que o tempo útil de conexão de cada usuário é extremamente curto: o suficiente para copiar (*download*) as notícias ou programas de interesse para seu disco.

O quadro de avisos central nada mais é que a memória do micro que o gerencia. Hoje, a maioria dos CBBS dispõe de discos rígidos de maior capacidade, mas pode-se operá-los só com dois acionadores de disquetes como memória auxiliar. Nesse caso, entretanto, a velocidade e a capacidade do sistema são significativamente diminuídas.

Muitos dos CBBS recentemente surgidos no Brasil constituem iniciativa de voluntários. Seguindo a tradição dos entusiastas da microinformática, eles são gratuitos (embora existam alguns, como o Sampa CBBS, em São Paulo, que se tornaram tão grandes que foi necessário fixar uma pequena taxa). Um dos sérios problemas desse sistema é impedir que pessoas não autorizadas tenham acesso a ele. Usualmente, isso é feito atribuindo-se uma senha a cada membro. Contudo, essa é uma questão difícil de ser solucionada, tanto que nem mesmo o Pentágono norte-americano vem obtendo sucesso no sentido de evitar que pessoas não autorizadas penetrem em seus monstrosos "mainframes". Na verdade, muitos CBBS recebem de braços abertos todos os intrusos.

PADRÕES

Evidentemente, a distância física entre o usuário e o CBBS não representa nenhum obstáculo, já que um computador pode perfeitamente se comunicar com outro, em qualquer parte do mundo, desde que ambos tenham modems compatíveis. Utilizando o equipamento correto, você poderá se beneficiar de mais de 1.500 serviços de troca de mensagens espalhados pelos EUA, Canadá e Europa. Em razão de não existirem padrões comuns de comunicação entre europeus e

■	O QUE É QUADRO DE AVISOS
■	CARREGANDO E DESCARREGANDO INFORMAÇÕES
■	PADRÕES TÉCNICOS
■	VELOCIDADE DE TRANSMISSÃO

■	O MELHOR MODEM
■	ACESSO ATRAVÉS DE MENUS
■	EXPLORANDO A REDE
■	TERMINOLOGIA ESPECIALIZADA

norte-americanos, o usuário que desejar contatar CBBS dessas regiões deverá dispor de modems diferentes.

O padrão adotado na Europa é o CCITT, sigla de uma agência da ONU denominada *Comité Consultatif International Téléphonique et Télégraphique*. Essa agência estabelece convenções internacionais para telecomunicações que podem também ser seguidas internamente

por um país-membro. Indiretamente, isso acaba facilitando a comunicação entre todos os proprietários de computadores. Cada país, porém, pode ter simultaneamente outras convenções internas, determinadas e aprovadas por órgãos estatais, como o Contel (Conselho de Telecomunicações), no Brasil.

Nos EUA, o chamado padrão Hayes é o mais comum para CBBS privados

(seu nome deriva da fábrica dos modems mais populares e baratos do país). O padrão Bell também é muito usado.

VELOCIDADES DE TRANSMISSÃO

As agências de normatização, nacionais e internacionais, deixam aos fabricantes de modems a criação de muitas



outras convenções. Os modems são classificados em função de diversos parâmetros operacionais e de desempenho como, por exemplo, a velocidade de transmissão. A grande maioria dos CBBS — os mais baratos — opera com modems de 300 bauds, isto é, eles recebem e transmitem dados à velocidade de 300 bits por segundo. (O nome da unidade de medida *baud* é uma homenagem ao engenheiro francês Baudot, considerado o inventor de um código para telex.) Isso equivale, aproximadamente, a um bit enviado a cada 3.3 milissegundos. Já os sistemas mais avançados de telecomunicações digitais seguem o padrão 1200/75 baud: o computador central transmite dados à velocidade de 1200 bauds e o usuário os envia a apenas 75 bauds. Sistemas como o Cirandão, da Embratel, e o Videotexto utilizam essa convenção (seu objetivo é reduzir o tempo de ocupação do computador central, devido ao custo elevado, e impor os gastos da interação ao usuário).

As taxas de transmissão normalmente são múltiplos de 300 (300, 600, 1200, 2400 etc.), duplicando progressivamente até atingir 9600 bauds, que é o máximo em uso atualmente.

Em muitos computadores, dependendo dos padrões definidos pelo CCICT para interfaces seriais RS-232, as voltagens de representação dos números binários 0 e 1 são, respectivamente, +12 e -12 V. Os micros que usam as voltagens +5 e 0 V necessitam de um conversor, que geralmente é barato.

BAUDS E BYTES

Cada caractere de informação transmitido por um modem compõe-se, basicamente, de um bit de início (*start bit*), dos bits de dados (*data bits*) de um bit de paridade (*parity bit*), usado em testes de erros de transmissão, e de um bit de término (*stop bit*). Como vemos, existem mais bits do que os oito de um byte; assim, para saber quantos bytes (caracteres de texto) são enviados por segundo, tomando como padrão uma taxa de 300 bauds, por exemplo, dividimos esse número por 11.

Embora não haja uma sequência "natural" nem uma quantidade preestabelecida para os diferentes bits que formam o caractere de transmissão, é importante que tanto o emissor quanto o receptor reconheçam a convenção seguida. Além disso, ambos devem ser capazes de formatar, enviar ou receber os dados conforme a taxa padronizada.

A velocidade adotada por um CBBS será escolhida levando-se em conta a ne-

cessidade de se reduzir os custos do serviço, até porque os *Computerized Bulletin Board Service* não dispõem dos recursos de hardware das grandes empresas de telecomunicações. O fato de serem empregados modems mais baratos implica o uso de uma taxa menor de transmissão a fim de evitar erros causados por interferências elétricas, normalmente filtradas por equipamentos sofisticados. Com um modem barato operando a altas taxas, qualquer pulso na rede poderia ser interpretado como um bit. Em velocidades pequenas, o pulso que define um bit é mais longo e, portanto, mais fácil de identificar.

De qualquer modo, o ruído será sempre um problema difícil de se resolver. Com a entrada em serviço das redes digitais de comunicações baseadas em fibras ópticas, espera-se que os sinais sejam mais "limpos" e rápidos.

DUPLEX

Denominamos *full-duplex* ou *half-duplex* ao parâmetro que varia de modem para modem e diz respeito à simultaneidade das funções de transmissão e recepção. Full-duplex significa que um determinado modem é capaz de receber e transmitir informações ao mesmo tempo; half-duplex indica que ele executa apenas uma função de cada vez.

Evidentemente, um modem half-duplex é mais barato e adequado a trabalhos com micros domésticos, ou até a algumas aplicações profissionais mais simples. Como é preciso aguardar uma transmissão terminar antes de iniciar a recepção (ou vice-versa), os erros cometidos na operação poderão trazer alguns inconvenientes. A possibilidade de se consumir um tempo excessivo praticamente desaconselha o uso deste equipamento para trabalhos profissionais.

Os CBBS se tornaram viáveis graças à sofisticação do hardware e software de comunicações. O maior passo nesse sentido foi a introdução de modems ou recursos de autodiscagem (*auto-dialing*, em inglês) e auto-resposta (*auto-answering*). A partir dessas inovações, as consultas ao computador hospedeiro do CBBS passaram a ser respondidas automaticamente com uma linha conectada entre receptor e transmissor.

SOFTWARE PARA COMUNICAÇÕES

Além do computador, do modem e do telefone, faz-se necessário um software especial que transforme o micro em um terminal de dados.

Classificamos o software de comunicações em: *burros e inteligentes*. O inteligente — essencial para quem deseja operar um CBBS — permite descarregar o programa (além de texto de mensagens) e armazená-lo automaticamente no disco. Já o software burro possibilita apenas a comunicação simples, sem qualquer forma de armazenamento ou listagem. Certos softwares inteligentes oferecem recursos extremamente úteis, entre eles o reconhecimento automático — através do programa — do protocolo seguido pelo computador hospedeiro e a armazenagem e discagem automática de vários números de CBBS.

A função básica do programa é a de permitir que o micro transmita e receba sinais (caracteres) pelo modem conectado ao sistema, mantendo o sincronismo entre o teclado, o vídeo e a memória. O software orienta o computador por intermédio de comandos relativos à forma com que os sinais serão transmitidos e recebidos.

Muitas pessoas não conseguem entender como dois computadores aparentemente incompatíveis podem se comunicar. E são essas mesmas pessoas que costumam criticar os fabricantes de micros por constantemente ignorarem os padrões que facilitam a interligação entre equipamentos distintos.

Por meio do CBBS, temos a falsa impressão de que um computador está identificando os comandos enviados por outro diferente (um Apple e um MSX, por exemplo). Na verdade, isso só acontece porque a maioria dos CBBS implementa seu software de acesso por meio de menus relativamente simples.

Com eles, o usuário seleciona uma opção pressionando uma tecla ou digitando uma linha de comando. Do outro lado, um programa especial "varre" o modem de recepção, de modo a captar os sinais enviados e a interpretar corretamente o pedido. A partir daí, os comandos adequados passam a ser acionados pelo computador hospedeiro.

EXPLORANDO OS CBBS

O primeiro passo para explorar todos os recursos dos CBBS à sua disposição é adquirir um modem adequado. Conforme abordamos, os CBBS normalmente adotam um tipo padrão de 300 bauds, com protocolo CCICT ou Hayes. Os sistemas comerciais, como o Videotexto e o Cirandão, utilizam o padrão 1200/75. Assim, o ideal é que o usuário compre um modem que tenha uma chave de seleção para cada modalidade.

O segundo passo consiste em provi-

denciar um software de comunicação (se ele já não tiver sido fornecido juntamente com o modem).

O terceiro passo é descobrir os números de telefone dos CBBS e seus detalhes técnicos de acesso. Esses dados podem ser encontrados em revistas especializadas de microcomputação.

Superadas todas as questões, passaremos à parte prática da operação. Conecte o modem à linha telefônica e ao microcomputador e execute o software de comunicação. Caso a versão utilizada disponha de autodiscagem, todo esse trabalho será feito automaticamente, uma vez especificado o CBBS que se deseja acessar. Do contrário, bastará ligar o número telefônico do mesmo e aguardar o sinal de recepção (geralmente um som agudo de poucos segundos denominado *portadora*). Recebido o aviso, pressione o botão que conecta o modem à linha (se o equipamento for de conexão direta), ou coloque o bocal do telefone nas "orelhas" de acoplamento (se for um modem acústico).

Para termos acesso a alguns CBBS, é necessário chamar o número desejado, deixar tocar uma vez, e, em seguida, discar novamente esse número. Só então o procedimento já descrito poderá ser observado.

Muitos CBBS funcionam de maneira semelhante. No início da operação, o usuário recebe uma mensagem que indica seu contato com o CBBS. Depois, deve enviar uma senha de entrada e/ou o número ou nome de identificação — principalmente se o uso do CBBS implica pagamento através de assinatura mensal ou anual. Tais sistemas, entretanto, chegam a oferecer, por um tempo limitado, algumas informações para convidados e "intrusos", isto é, pessoas ainda não registradas no serviço.

Os CBBS gratuitos, dependendo do tipo, podem solicitar seu nome e cidade ou endereço e número telefônico, bem como algumas especificações do micro usado (modelo, tamanho da tela etc.). Esses detalhes são importantes para que o computador hospedeiro saiba como operar durante a comunicação.

Na fase seguinte, você receberá informações do CBBS: horários de funcionamento, limite de tempo para cada chamada etc. Tais dados interessam, sobretudo, aos CBBS voluntários, já que algumas chamadas são feitas em horários inconvenientes (desagradando o operador, que geralmente utiliza o telefone de sua própria casa para isso).

Assim como o videotexto, a maioria dos CBBS é operada através de menus. A escolha de uma opção em um menu poderá levar a menus secundários e ter-

ciários etc. ou à execução da opção. Muitos CBBS possuem uma seção que permite aos novos usuários registrar seus dados (endereço, nome, telefone e senha de acesso). Outras opções oferecidas podem incluir uma seção técnica, comunicações com o gerente etc.

O elemento fundamental do CBBS está no quadro que contém avisos e notas para conhecimento geral. Isso, porém, não exclui a possibilidade de existir também uma seção de mensagens pessoais: "caixas postais", com acesso limitado por senhas secretas. Esse sistema permite deixar informações reservadas para outros usuários ou formar subquadros de avisos dentro do CBBS.

TERMINOLOGIA

Há vários termos técnicos que, embora comuns no campo da comunicação entre computadores, são pouco conhecidos em outros ramos da informática. Devido a isso, às vezes, cria-se uma certa confusão sobre seu significado.

A seguir, uma lista dos termos que normalmente são mal interpretados:

Videotexto - Sistema especial de comunicação entre computadores, oferecido como um serviço público pelas companhias telefônicas. Através de uma linha comum, o computador central envia ao usuário a informação (organizada em páginas ou telas). Esta é recebida por um terminal especial de videotexto ou um micro qualquer, que podem responder fazendo novas solicitações de dados. No Brasil, tem-se acesso aos sistemas de videotexto por meio de uma assinatura mensal junto à companhia telefônica de cada cidade. Na Europa, esse serviço, implantado em vários países, é denominado *Prestel*.

Teletexto - Sistema semelhante e mais barato que o videotexto. Aqui, entretanto, a informação é enviada apenas em um sentido (do computador central para o terminal), aproveitando o intervalo disponível entre dois quadros exibidos pela TV. Este sistema não existe no Brasil. Exemplos no exterior são os CEEFAX e o ORACLE, utilizados na Inglaterra. Aparelhos de TV de algumas marcas já são fabricados com a interface embutida.

Videotex - Termo geral que engloba o videotexto, o teletexto e os CBBS. Criou-se uma certa confusão depois que alguns CBBS evoluíram e se transformaram em uma rede, apesar de conservarem o mesmo nome.

Rede - Em sua forma mais restrita, refere-se à ligação direta entre vários computadores com objetivos específicos (normalmente para fins comerciais ou de pesquisa, como a rede brasileira RENPAC). As redes dividem-se em dois tipos: a local (LAN, ou *Local Area Network*), que, geralmente, envolve apenas a ligação direta entre micros e seus periféricos ou a um computador central, e a telemática (WAN, ou *Wide Area Network*), que inclui ligações remotas, via rede telefônica, microondas ou satélites. Através da LAN, os usuários podem compartilhar recursos como discos e impressoras. A WAN, por sua vez, permite a intercomunicação e a consequente troca de mensagens e transações (como em uma rede de automação bancária, que liga várias agências aos computadores centrais).

Uma rede pode ser privada ou pública. Existem vários exemplos de redes públicas, como o Cirandão, da Embratel, ou o CompuServe, nos EUA, com mais de 250.000 usuários.

Nos últimos anos, o termo rede vem perdendo sua especificidade, passando a significar qualquer forma de intercomunicação realizada entre computadores. Por esse conceito, todos os serviços do tipo videotex também poderiam ser considerados uma rede.

Telex e teletex - Serviços em fase de integração a redes de computadores e que, por isso, ainda são confundidos. O telex é um sistema telefônico de comunicação de textos, de baixa velocidade (15 bauds), gerenciado por uma central especial computadorizada. Existem interfaces que permitem dar a um micro a função de um terminal de telex (daí serem chamadas de microtelex).

O teletex é uma evolução técnica do telex e, muito provavelmente, seu substituto direto. Utiliza o mesmo sistema de comunicação, porém, a uma velocidade bem maior, de modo a facilitar o uso de micros como terminais. Ainda não foi implantado no Brasil.

Correio eletrônico - Serviço oferecido por diversos tipos de rede, videotexto ou CBBS. Consiste, basicamente, de "caixas postais" (espaço reservado no disco central a cada assinante), capazes de receber mensagens individuais ou circulares, enviadas por outros usuários. A comunicação é feita a nível privado, e protegida por senhas. Em alguns países, o sistema é prerrogativa do serviço estatal de correios. No Brasil, como em outras nações, esse monopólio já foi rompido (Cirandão, Mensagem, Videotexto).

UM EDITOR MUSICAL (2)

Apresentamos aqui a segunda parte de nosso editor musical. Carregue a primeira parte da fita cassete ou disco e digite, na sequência, a listagem dada abaixo. Depois, grave as duas partes para juntá-las à que iremos publicar no próximo e último artigo da série — onde você encontrará, também, instruções detalhadas para uma eficiente utilização do programa.

Como você terá oportunidade de observar, à medida que for digitando, o programa é composto por várias sub-rotinas chamadas do menu principal. É possível ter uma idéia da função de cada uma delas lendo os itens do menu — que pode ser exibido na tela ainda que o programa esteja incompleto.

S

```
2000 CLS
2010 GOSUB 2500
2140 PRINT 'ct; Notas tecladas
      (';maxnotes;'Max.)'''
2160 INPUT "Entre Notas - <RET>
      para acabar";NS
2175 IF LEN (NS)=0 THEN GOTO 1
90
2180 FOR i=1 TO LEN (NS): IF (N
      S(i)<"0" OR NS(i)>"9") AND (NS(
      i)<>"j") THEN GOTO 2000
2190 LET N=VAL (NS)
2200 IF INT (N/1000)>11 AND INT
      (N/1000)<>-4 THEN GOTO 2000
2210 IF N<0 THEN GOTO 2280
2220 LET M=INT (N/100): LET D=N
      -M*100
2230 LET O=M-INT (M/10)*10: IF
      O<1 OR O>7 THEN GOTO 2000
2240 LET M=INT (M/10)+(O-1)*12-
      36
2260 LET ct=ct+1: LET t(2*ct-1)
      =D: LET t(2*ct)=M
2270 GOTO 2000
2280 LET M=-4: LET D=0-(N-M*100
      )
2290 IF M<>-4 THEN GOTO 2000
2310 GOTO 2260
2500 PRINT "Entre a Nota na for
      ma <Numero>, <Oitava>, <Duracao>."
2520 PRINT "C - 0   E - 4   G# -
      8" "C# - 1   F - 5   A - 9" "D
      - 2   F# - 6   A# - 10" "D# - 3
      G - 7   B - 11"
2560 PRINT "'PAUSA e -4'" "Semi
      colcheia=1   Minima = 8" "Col
      cheia =2   Semibreve=16" "Se
```

```
minima =4   Oitava =1 a 7"
2600 PRINT "'Duracao DEVE ter 2
      digitos" "ex. 3304-D#, Colchei
      a 3a. oitava" "ex. 6408-F#, Min
      ima 4a. oitava"
2630 RETURN
3000 CLS
3010 PRINT "Tocar musica"
3020 PRINT : PRINT : PRINT
3030 PRINT "Digite Tempo - (1-
      15) "
3040 INPUT S
3050 IF S<1 OR S>15 THEN GOTO
      3000
3060 LET tempo=0.02*(16-S)
3070 FOR i=1 TO ct
3080 LET D=t(2*i-1): LET M=t(2*
      i)
3090 IF m=-4 THEN GOTO 3120
3100 SOUND D*tempo,M
3110 GOTO 3130
3120 PAUSE 50*D*tempo
3130 NEXT i
3140 RETURN
4000 CLS
4010 PRINT "EDITOR MUSICAL" ""
      "D - Mostrar todas as notas" ""
      "E - Editar uma nota" "" "I - Inse
      rir uma nota" "" "X - Apagar uma
      nota" "" "R - Retornar ao menu
      principal"
4090 PRINT "'Escolha uma opca
      o - "
4100 LET OS=INKEY$: IF OS="" TH
      EN GOTO 4100
4105 IF CODE (OS)<97 THEN LET
      OS=CHR$(CODE (OS)+32)
4110 IF OS<>"d" AND OS<>"e" AND
      OS<>"i" AND OS<>"r" AND OS<>"x
      " THEN GOTO 4000
4120 IF OS="r" THEN RETURN
4130 IF OS="e" THEN GOTO 4300
4134 IF OS="i" THEN GOTO 4700
4136 IF OS="x" THEN GOTO 4800
4140 CLS
4150 FOR i=1 TO ct
4160 LET M=t(2*i): LET D=t(2*i-
      1)
4170 LET O=INT ((M+36)/12)+1
4180 LET N=(M+36)-(O-1)*12
4190 PRINT i:" Nota- ";N;" Oit.
      - ";O;" Dur.- ";D
4195 POKE 23692,255
4200 IF i=20*INT (i/20) THEN G
      OTO 4220
4210 GOTO 4250
4220 PRINT "'Qualquer tecla par
      a continuar "
4230 PAUSE 0
4240 PRINT
4250 NEXT i
```

Continue digitando nosso programa editor de melodias. Com as listagens fornecidas no próximo artigo da série, ele ficará completo e você poderá dar vazão a todo o seu talento musical.



```
650 P=P-36
660 IF P=1ANDC<6THEN C=C+1:OCS=
      MIDS(STR$(C),2)
670 IF P=2ANDC>1THEN C=C-1:OCS=
      MIDS(STR$(C),2)
680 IF (P=3ORP=6)AND LE>1 THEN L
      E=LE-1:LES=MIDS(R1$,LE,1)+" "
690 IF (P=4ORP=7)AND LE<7THEN LE
```



SEGUNDA PARTE
DO PROGRAMA EDITOR
GRAVAÇÃO E LEITURA
DIGITAÇÃO DAS
SUB-ROTINAS

```
=LE+1:LES=MID$(R1$,LE,1)+" "
700 IF P=5 THEN I$="p":P$="":GO
TO 750
710 IF P=6ORP=7 THEN LES=LEFTS(
LES,1)+" "
720 IF P=8 THEN RETURN
730 IF P=9 AND NN>0 THEN NN=NN-
1
740 GOTO 510
750 U$="L":IF I$>"a" AND I$<="
q" THEN P$=CHR$(ASC(I$)-32) ELS
E IF I$<>"p" THEN P$=I$+"#"ELSE
P$="":U$="R"
760 PLS="V15T"+STR$(TE)+"O"+OCS
+U$+L2$(INSTR(R1$,LEFTS(LES,1))
)
770 IF MID$(LES,2,1)="-." THEN P$
=P$+"."
```

```
780 PLS=PLS+P$:PLAY PLS
790 NN=NN+1:N$(NN)=I$+OCS+LES
800 GOTO 580
810 CLS:COLOR 4,15
820 LOCATE 3,2:PRINT"ENTRADA MA
NUAL DE NOTAS";
830 LOCATE 3,18:PRINT"'m' RETO
RNAR-MENU PRINCIPAL"
840 LOCATE 3,19:PRINT"USE FORMA
TO:alh.('.'opcional)
850 GOSUB 360:LOCATE 3,21:PRINT
"ENTRE STRING NOTA:";
860 IF NN=MX THEN LOCATE 1,23:P
RINT"MAXIMO NUMERO DE NOTAS COL
OCADO!":FOR S=1TO1000:NEXT:RETU
RN
870 LINE INPUT AS
880 IF AS="" THEN 870
```

```
890 IF AS="m"OR AS="M" THEN RET
URN
900 IF LEN(AS)>4ORLEN(AS)<3 THE
N 970
910 IF (INSTR(R1$,LEFTS(AS,1)))
=0 THEN 970
920 IF LEFTS(AS,1)="p" THEN AS=
"p"+" "+MID$(AS,3):GOTO 940
930 IF (INSTR(R2$,MID$(AS,2,1))
)=0 THEN 970
940 IF (INSTR(R1$,MID$(AS,3,1))
)=0 THEN 970
950 IF MID$(AS,4,1)<>"." THEN A
$=LEFT$(AS,3)+" "
960 GOTO 990
970 LOCATE 21,21:PRINT"ENTRADA
ILEGAL!";CHR$(7)
980 LOCATE 21,21:PRINT"
":GOTO 850
990 NN=NN+1:N$(NN)=AS
1000 GOTO 850
1010 IF NN=0 THEN RETURN
1020 COLOR 1,15:LOCATE 0,0:PRIN
T "MODO EDIÇÃO- 1/2/3/4"
```

```
1030 LOCATE 0,1:PRINT STRINGS(4
0,32)
1040 LOCATE 0,2:PRINT STRINGS(4
0,32)
1050 LOCATE 0,18:PRINT STRINGS(
40,32)
1060 LOCATE 0,3:PRINT STRINGS(4
0,32)
1070 LOCATE 2,20:PRINT"1:APAGAR
NOTAS";
1080 LOCATE 2,21:PRINT"2:INSERI
R NOTAS";
1090 LOCATE 2,22:PRINT"3:ALTERA
R NOTAS";
1100 LOCATE 2,23:PRINT"4:CONTIN
UA";
1110 AS=INKEY$:IF AS<"1"ORAS>"4
" THEN 1110
1120 OP=VAL(AS)
1130 ON OP GOTO 1150,1250,1410,
1140
1140 RETURN
1150 LOCATE 12,0:PRINT"1:APAGAR
NOTAS"
1160 LOCATE 0,2:INPUT"INICIA NA
NOTA";ST
1170 IF ST=0 THEN 1010
1180 IF ST>NN THEN 1150
1190 LOCATE 0,3:INPUT"QUANTAS A
PAGA (ENTER=1)";ND
1200 IF ND<=0 THEN ND=1
1210 IF ST+ND-1>NN THEN ND=NN-S
T+1
1220 FOR K=1TOND
1230 FOR J=ST TO NN-1:N$(J)=N$(
J+1):NEXT
1240 NN=NN-1:NEXT K:I=SL-1:GOTO
```




```

1010
1250 LOCATE 12,0:PRINT"2:INSERIR NOTAS"
1260 LOCATE 0,2:INPUT"INICIA INSERCAO APOS QUE NOTA";ST
1270 IF ST<0 THEN 1010
1280 IF ST>NN THEN ST=NN
1290 LOCATE 0,3:PRINT"TECLE'M'PARA ENCERRAR:";
1300 IF NN=MX THEN LOCATE 2,18:PRINT"MAXIMO NUMERO DE NOTAS COLOCADAS!";FOR D=1TO 2000:NEXT:I=SL-1:GOTO 1010
1310 LINE INPUT NS:IF NS="M"OR NS="m"THEN I=SL-1:GOTO 1010
1320 IF LEN(NS)<3 THEN LOCATE 2,3:PRINT"ILEGAL":GOTO 1290
1330 IF INSTR(R3$,LEFT$(NS,1))=0ORINSTR(R2$,MIDS(NS,2,1))=0ORINSTR(R1$,MIDS(NS,3,1))=0 THEN LOCATE 22,3:PRINT"ILEGAL":GOTO 1290 NOTAS
1340 IF MIDS(NS,4,1)=". "THEN NS=LEFT$(NS,4)ELSE NS=LEFT$(NS,3)+" "
1350 IF ST=NN THEN 1380
1360 FOR K=NN+1 TO ST+2 STEP -1
1370 NS(K)=NS(K-1):NEXT
1380 NS(ST+1)=NS
1390 ST=ST+1:NN=NN+1
1400 GOTO 1290
1410 LOCATE 13,0:INPUT"3:ALTERAR NOTA";ST
1420 IF ST=0 THEN 1010
1430 IF ST>NN THEN ST=NN
1440 LOCATE 11,2:PRINT NS(ST)
1450 RT=255:LP=-2:I=ST:GOSUB 1620
1460 LOCATE 0,2:PRINT"NOVO VALOR:";LINE INPUT NS
1470 IF LEN(NS)<3 THEN 1460
1480 IF INSTR(R3$,LEFT$(NS,1))=0ORINSTR(R2$,MIDS(NS,2,1))=0ORINSTR(R1$,MIDS(NS,3,1))=0 THEN 1460
1490 IF MIDS(NS,4,1)=". "THEN NS=LEFT$(NS,4)ELSE NS=LEFT$(NS,3)+" "
1500 NS(ST)=NS:GOSUB 1620 :FOR K=1TO2000:NEXT:RT=0:I=SL-1:LP=0:GOTO 1010
1510 IF NN=0 THEN RETURN ELSE CLS:COLOR1,15:LOCATE 11,1:PRINT"LISTAR NOTAS"
1520 C=1:SL=1:EL=NN:PS="N":RT=0
1530 LOCATE 4,3:INPUT"LISTAR NA IMPRESSORA(S/N)";PS
1540 IF PS="S" THEN C=2
1550 LOCATE 4,5:INPUT"INICIO EM (ENTER= 1)";SL
1560 IF SL<=0 THEN SL=NN:EL=NN
1570 LOCATE 4,6:INPUT"FINAL EM (ENTER=FIM)";EL
1580 IF EL=0 OR EL>NN THEN EL=NN
1590 IF SL>EL THEN SL=EL
1600 CLS:LP=0

```

```

MIDS(STR$(C),2)
620 IF P=2ANDC>1THEN C=C-1:OCS=MIDS(STR$(C),2)
630 IF (P=3ORP=6)AND LE>1 THEN LE=LE-1:LES=MIDS(R1$,LE,1)+" "
640 IF (P=4ORP=7)AND LE<5 THEN LE=LE+1:LES=MIDS(R1$,LE,1)+" "
650 IF P=5 THEN IS="p":PS="":GO TO 700
660 IF P=6ORP=7 THEN LES=LEFT$(LES,1)+" "
670 IF P=8 THEN RETURN
680 IF P=9 AND NN>0 THEN NN=NN-1
690 GOTO 490
700 IF IS>="a" AND IS<="q" THEN PS=CHR$(ASC(IS)-32) ELSEIF IS<>"p" THEN PS=IS+"#" ELSE PS=""
710 PLS="V31:T"+STR$(TE)+"L"+L2$(INSTR(R1$,LEFT$(LES,1)))
720 IF MIDS(LES,2,1)=". " THEN PLS=PLS+" "
730 PLS=PLS+"O"+OCS+PS:PLAY PLS
740 NN=NN+1:NS(NN)=IS+OCS+LES
750 GOTO 530
760 CLS
770 PRINT@4,"ENTRADA MANUAL DE NOTAS":PRINT@33,"ENTRE'm'PARA RETORNAR AO MENU"
780 PRINT"USE FORMATO:'alh.'('OPCIONAL)"
790 GOSUB 280:PRINT@416,"ENTRE STRING NOTA:"
800 IF NN=MX THEN PRINT@448,"MAXIMO NUMERO DE NOTAS COLOCADO!":FOR D=1TO1000:NEXT:RETURN
810 POKE 282,0:PRINT@440:PRINT@462,;:LINE INPUT AS
820 IF AS="" THEN 810
830 IF AS="m"OR AS="M"THEN RETURN
840 IF LEN(AS)>4OR LEN(AS)<3 THEN 910
850 IF (INSTR(R3$,LEFT$(AS,1)))=0 THEN 910
860 IF LEFT$(AS,1)="p" THEN AS="p"+" "+MIDS(AS,3):GOTO 880
870 IF (INSTR(R2$,MIDS(AS,2,1)))=0 THEN 910
880 IF (INSTR(R1$,MIDS(AS,3,1)))=0 THEN 910
890 IF MIDS(AS,4,1)<>". " THEN AS=LEFT$(AS,3)+" "
900 GOTO 920
910 PRINT@448,LEFT$(AS,4);"-ENTRADA ILEGAL!":SOUND 1,5:GOTO 810
920 NN=NN+1:NS(NN)=AS
930 GOTO 790
940 IF NN=0 THEN RETURN ELSE POKE 282,0
950 PRINT@448,"1:APAGAR NOTA ", "2:INSERIR NOTAS", "3:ALTERAR NOTAS ", "4:CONTINUAR";
960 AS=INKEY$:IF AS<"1"OR AS>"4" THEN 960
970 OP=VAL(AS)
980 ON OP GOTO 1000,1100,1260,990
990 RETURN
1000 IF NN=0 THEN 940 ELSE CLS:INPUT"INICIA NA NOTA":ST
1010 IF ST=0 THEN 940

```

```

1020 IF ST>NN THEN 1000
1030 PRINT@64,"QUANTAS APAGAS(ENTER=1)";:INPUT ND
1040 IF ND<=0 THEN ND=1
1050 IF ST+ND-1>NN THEN ND=NN-ST+1
1060 FOR I=1 TO ND
1070 FOR J=ST TO NN-1:NS(J)=NS(J+1):NEXT
1080 NN=NN-1
1090 NEXT I:RETURN
1100 CLS:PRINT@7,"MODO INSERCAO DE NOTAS"
1110 PRINT@64,"INICIA INSERCAO APOS QUE NOTA":INPUT ST
1120 IF ST<0 THEN 940
1130 IF ST>NN THEN ST=NN
1140 PRINT@64,"TECLE'm'QUANDO VOCE TERMINAR":PRINT
1150 IF NN=MX THEN PRINT"MAXIMO NUMERO DE NOTAS COLOCADO!":FOR D=1 TO 1000:NEXT:RETURN
1160 INPUT NS:IF NS="M"OR NS="m" THEN RETURN
1170 IF LEN(NS)<3 THEN PRINT"ILEGAL":GOTO 1160
1180 IF INSTR(R3$,LEFT$(NS,1))=0ORINSTR(R2$,MIDS(NS,2,1))=0ORINSTR(R1$,MIDS(NS,3,1))=0 THEN PRINT"ILEGAL":GOTO 1160
1190 IF MIDS(NS,4,1)=". "THEN NS=LEFT$(NS,4)ELSE NS=LEFT$(NS,3)+" "
1200 IF ST=NN THEN 1230
1210 FOR I=NN+1 TO ST+2 STEP -1
1220 NS(I)=NS(I-1):NEXT
1230 NS(ST+1)=NS
1240 ST=ST+1:NN=NN+1
1250 GOTO 1150
1260 CLS:PRINT"MUDAR QUE NOTA":INPUT ST
1270 IF ST=0 THEN 940
1280 IF ST>NN THEN ST=NN
1290 PRINT:PRINT"VALOR ATUAL:"AS=NS(ST):RT=255:GOSUB 350
1300 PRINT NS(ST)
1310 PRINT@320:PRINT@320,"";:INPUT"NOVO VALOR:";NS
1320 IF LEN(NS)<3 THEN 1310
1330 IF INSTR(R3$,LEFT$(NS,1))=0ORINSTR(R2$,MIDS(NS,2,1))=0ORINSTR(R1$,MIDS(NS,3,1))=0 THEN 1310
1340 IF MIDS(NS,4,1)=". " THEN NS=LEFT$(NS,4)ELSE NS=LEFT$(NS,3)+" "
1350 NS(ST)=NS:FOR K=1TO2000:NEXT:GOTO940
1360 IF NN=0 THEN RETURN ELSE CLS:PRINT@7,"OPCAO LISTAR NOTAS".C=0
1370 IF(PEEK(65534)AND1)=1 THEN 1400
1380 POKE 282,255:PRINT@64,"";:INPUT"LISTAR NA IMPRESSORA(S/N)";PS
1390 IF PS="S" THEN C=-2:POKE 150,1
1400 PRINT@128,"INICIO NA NOTA":INPUT ST
1410 IF ST<=0 THEN ST=NN
1420 IF ST>NN THEN ST=NN
1430 CLS:LP=0

```





OS SEGREDOS DO TRS-80 (5)

Nos artigos anteriores desta série, examinamos os recursos "secrets" do micro TRS-80 para a saída de vídeo. Entre outras coisas, vimos como gerar pseudo-sprites e como levar cópias de telas para a memória e vice-versa.

O teclado do TRS-80 também tem várias características interessantes, que não estão bem documentadas nos manuais de operação ou mesmo nos livros mais elementares sobre o assunto. Neste artigo, você aprenderá alguns truques que podem ser executados com facilidade, como o bloqueio da tecla <BREAK>, a implementação de um cursor piscante, a programação da repetição automática de teclas e a entrada direta de símbolos gráficos pelo teclado.

Esses recursos são possíveis porque a memória do TRS-80 contém o mapa do teclado, assim como o do vídeo. O teclado desse micro é do tipo matricial, ou seja, ao se pressionar uma tecla, ela aciona o cruzamento entre dois condutores: um colocado nas colunas, e outro, nas linhas. Isso gera um código binário, que é recuperado pelo software processador do teclado. Este efetua uma varredura a intervalos de alguns milissegundos, passando por todos os fios horizontais e verticais. O número obtido é depositado em uma localização específica da memória de teclado.

Diversas outras localidades da memória RAM reservada para a área de trabalho do interpretador BASIC são dedicadas ao teclado. Através de comandos PEEK e POKE, é possível verificar, e mesmo alterar, seu conteúdo, produzindo curiosos e variados efeitos.

BLOQUEIO DA TECLA <BREAK>

Como você já sabe, a tecla <BREAK> do TRS-80 interrompe um programa BASIC em execução, independentemente do que ele estiver fazendo, e retorna o controle ao interpretador.

A facilidade de interromper o programa pode ser um inconveniente quando se deseja torná-lo imune a erros de operação ou se quer impedir que alguém o liste. Imagine a seguinte situação: você colocou um programa educativo nas mãos de uma criança; ao aparecer a

mensagem "PRESSIONE QUALQUER TECLA PARA CONTINUAR", ela aciona justamente a tecla <BREAK>! A culpa, com certeza, é sua...

Para evitar ocorrências desse tipo, convém bloquear o efeito da tecla <BREAK> através de comandos POKE nas localidades de memória 16396 e 16397:

```
POKE 16396,175:POKE 16397,201
```

Para ativar novamente a tecla, basta utilizar o comando:

```
POKE 16396,201
```

Esse truque funciona em qualquer micro compatível com os modelos I e III da linha TRS-80, e pode ser empregado tanto como comando direto quanto dentro de um programa.

Há um problema, entretanto. Se você estiver usando o BASIC de disco, e tentar acessá-lo durante a desativação da tecla <BREAK>, o computador poderá "congelar". Para evitar que isso aconteça, aconselha-se recorrer a outra localização de memória para dar o comando POKE. Não se esqueça, porém, de que esta varia de acordo com o sistema operacional de disco (DOS) empregado:

DOS	DESATIVA	ATIVA
TRSDOS 2.3	POKE 23886,0	POKE 23886,1
NEWDOS 2.1	POKE 23461,0	POKE 23461,1
NEWDOS 80	POKE 19408,0	POKE 19408,1

A maioria dos micros nacionais da linha TRS-80 utiliza sistemas compatíveis com o TRSDOS (pronuncia-se *trisdós*). Se seu micro usa a versão CP/M, você não poderá desativar a tecla <BREAK> por meio da operação sugerida.

Um aviso final: se seu programa ainda não foi gravado, tenha muito cuidado ao usar esse truque, assegurando que a tecla <BREAK> possa ser reativada. Caso contrário, você poderá ficar na desagradável situação de nunca mais poder listar seu próprio programa!

AUTO-REPETIÇÃO

Os modelos da linha TRS-80 I e III (como o Prológica CP-500) não têm a capacidade de repetição automática das teclas — ou seja, as teclas não se auto-

repetem durante o período em que estão sendo pressionadas. Este é um recurso útil para uma série de aplicações: por exemplo, jogos em que o movimento de um cursor ou o disparo de uma metralhadora são controlados por teclas auto-repetitivas. Há uma maneira, porém, de saber se uma tecla continua sendo pressionada ou não, o que nos permite implementar uma rotina de repetição. Se o valor dado por um PEEK(14591) for igual a 0, nenhuma tecla está sendo pressionada; se esse valor for maior que 0, uma tecla está sendo pressionada.

Para entender como funcionaria um laço simples de repetição, digite:

```
10 PRINT PEEK(14591)::GOTO 10
```

Ao ser executado, o programa imprime uma sequência de zeros na tela. Quando uma tecla é pressionada, esse valor muda. Note que cada tecla ou combinação de teclas (pressionadas juntas) tem um valor diferente, que não corresponde ao valor ASCII da tecla.

Para aplicar esse expediente, identifique os números das teclas que deseja usar para direcionar a lógica de determinado programa e introduza vários IF dentro do mesmo. Se você quiser utilizar os valores ASCII gerados por cada tecla pressionada, precisará de um IF para testar se a tecla continua pressionada, e outro para localizar essa tecla.

Como exemplo, execute este programa. Ele desloca um cursor pela tela, sob o controle de quatro teclas (flechas).

```
10 CLS:X=64:Y=24
15 SET(X,Y)
20 IF T%>0 AND PEEK(14591)>0 THEN 50
25 TS=INKEY$:IF TS="" THEN 25
30 T%=ASC(TS)
50 IF T%=8 THEN X=X-1:GOTO 15
60 IF T%=9 THEN X=X+1:GOTO 15
70 IF T%=91 THEN Y=Y-1:GOTO 15
80 IF T%=10 THEN Y=Y+1:GOTO 15
90 GOTO 25
```

A linha 10 do programa define a posição inicial do cursor (um pequeno ponto na tela), "aceso" pelo comando SET, com as coordenadas X e Y na tela.

A linha 20 verifica se alguma tecla está sendo pressionada — ou seja, se o valor T% (código ASCII da tecla, deter-

minado pelo comando INKEY\$) for maior que zero. Se for, a linha 25 atribui o código ASCII da tecla pressionada a T%, e a linha 30 converte esse código para decimal. As linhas 50 a 80 verificam se T% é igual a 8, 9, 91 ou 10, que correspondem às teclas de seta esquerda, direita, cima e baixo, respectivamente. Se for, as linhas 50 a 80 alteram as coordenadas X e Y do cursor e saltam para a linha 15, onde o cursor é movido para a nova posição.

Se nenhuma tecla estiver sendo pressionada, o programa salta diretamente para a linha 90, onde o cursor é movido para a posição inicial.

Este programa é apenas um exemplo de como utilizar o recurso de repetição automática das teclas. Você pode adaptá-lo para qualquer aplicação que exija a repetição automática de uma tecla.

■	TRUQUES COM O TECLADO
■	COMO DESATIVAR A TECLA <BREAK>
■	ROTINA PARA A REPETIÇÃO AUTOMÁTICA

■	DAS TECLAS CURSOR DE TEXTO PISCANTE
■	ENTRADA DIRETA DE SÍMBOLOS GRÁFICOS

minado na linha 30) e o conteúdo da memória 14591 são, simultaneamente, maiores que 0. Em caso afirmativo, o programa pula para a linha 50, que inicia vários testes para identificar qual das teclas com flecha foi pressionada. As coordenadas X e Y são alteradas para mais ou para menos, e um outro ponto é aceso na tela na nova posição. A linha 20 testa mais uma vez se essa tecla continua pressionada, executa um deslocamento e assim por diante.

Se nenhuma tecla estiver sendo pressionada, o programa vai para a linha 25, que cria um laço de varredura do teclado com a função **INKEY\$**. Essa linha se repete até que se pressione alguma tecla. Então, o valor ASCII da tecla é identificado e armazenado em **T%**, pela linha 30. A partir daí, o micro se comporta como se tivesse a capacidade de repetição automática das teclas.

Observe que, se nenhuma das quatro teclas com flecha tiver sido pressionada, o programa retorna ao laço de espera situado na linha 25.

Ao executar esse programa, tenha o cuidado de não ultrapassar os limites da tela com o cursor, para que não ocorra um erro na linha 15. Se quiser melhorar o programa, introduza testes para as alterações de X e Y, de modo a impedir que eles ultrapassem os valores mínimo e máximo da tela.

O CURSOR PISCANTE

Normalmente, o cursor de texto do TRS-80 é um traço de sublinhar, não piscante. Não é fácil localizá-lo com a tela cheia de texto. A dificuldade aumenta em aplicações de deslocamento bidimensional do cursor, sobretudo se ele se sobrepõe a um trecho gráfico.

Um bom recurso consiste em fazer o cursor piscar repetidamente, distinguindo-se dos caracteres de fundo. No TRS-80, porém, só se obtém esse efeito por meio de software, já que nenhum de seus comandos permite alterar a função do cursor. Portanto, a rotina precisa ser suficientemente rápida para não "segurar" o usuário que está digitando o texto a uma certa velocidade.

Um problema adicional é fazer o cur-

sor piscar quando está sobre um caractere já impresso. Nesse caso, devemos copiar o valor do caractere em uma variável, antes de piscar, e recolocá-lo na tela. Os comandos **PEEK** e **POKE** desta rotina encarregam-se disso:

```
100 T$=INKEY$: IF T$<>" " THEN
120
110 POKE X%.95:POKE X%,C%:GOTO
100
120 RETURN
```

Antes de chamar a rotina, coloque a posição atual do cursor em **X%**. Este é um valor entre 15360 e 16383, números que correspondem às locações da memória de vídeo na RAM. Para obtê-lo, examina-se o conteúdo das memórias 16416 e 16417. O valor ASCII do caractere nessa posição deve ser colocado em **C%**. Assim, a rotina poderá fazer o cursor piscar, após verificar o código armazenado na locação **X%** de vídeo:

```
10 X%=PEEK(16417)*256+PEEK
(16416)
20 C%=PEEK(X%)
30 GOSUB 100:PRINT T$;:GOTO 10
```

A linha 30 chama a rotina e imprime o caractere que se pressionou, retornando pela sub-rotina em **T\$**. A volta à linha 10 tem o efeito de deslocar o cursor uma posição à direita no vídeo, reiniciando a rotina de piscar o cursor.

Essa rotina permite ainda que se modifique a forma do cursor. Note que usamos o código ASCII 95, que corresponde ao traço de sublinhar. Para colocar na tela um retângulo sólido, substitua 95 por 191, na linha 110.

GRÁFICOS PELO TECLADO

Como vimos em artigo anterior (página 1259), o emprego da rotina **INKEY\$** no lugar do comando **INPUT** possibilita a entrada dos caracteres gráficos do TRS-80 diretamente pelo teclado. Assim, se usarmos uma combinação das teclas **<SHIFT>** e **<LINEFEED>** (tecla para baixo), teremos o mesmo efeito da tecla **<CONTROL>** de outros computadores, e poderemos gerar códigos ASCII diferentes, pelo teclado. Se pressionarmos, por exemplo, a tecla A, juntamente

MICRO DICAS

VARREDURA DO TECLADO

A função **INKEY\$** é bastante útil para se detectar, dentro de um programa, se alguma tecla foi pressionada. Mas, às vezes, ela não é suficiente para satisfazer certas necessidades do programador — como indicar se duas ou mais teclas foram pressionadas simultaneamente, ou detectar o acionamento de uma tecla que não gera um código através do **INKEY\$**, como **<SHIFT>**.

Conhecendo a organização do teclado, porém, esse tipo de verificação não oferece dificuldade. O teclado é disposto na forma de uma matriz. Quando uma tecla é pressionada, um determinado valor numérico entre 1 e 128 (potências de 2) é depositado na memória RAM correspondente a uma fileira (um dos seguintes endereços: 14337, 14338, 14340, 14344, 14352, 14368, 14400 ou 14464). A correspondência entre tecla e endereço nos permite identificar a tecla ou teclas pressionadas. Por exemplo, a tecla **<SHIFT>**, quando pressionada, gera o código 1 na locação de memória 14464.

Quando duas ou mais teclas são pressionadas simultaneamente, gerando códigos no mesmo endereço, o número resultante é a soma dos códigos individuais. Assim, se J, K e L forem pressionadas, o número $4 + 8 + 16 = 28$ será colocado no endereço 14338.

Para montar sua própria tabela de correspondência entre teclas e valores, faça um programa de uma linha, usando um **PEEK** para examinar o conteúdo de cada uma das memórias dentro de um laço infinito.

com as anteriores, geramos o código ASCII 1. Detectando esse código na rotina **INKEY\$**, bastará somá-lo ao valor 128 para obtermos em **T\$** (variável de saída) o gráfico correspondente, que também será impresso na tela.

Substitua a linha 130 por:

```
130 IF ASC(T$)<32 THEN T$=CHR$(
T$+128)
140 RETURN
```


COMPRESSÃO DE TEXTOS (2)

Neste artigo, examinaremos novas técnicas de compressão de textos. Com as várias alternativas dadas, não será difícil reduzir o espaço de memória ocupado por seu jogo.

Como vimos no artigo da página 1332, podemos conseguir um alto índice de compressão de textos através da utilização de um conjunto reduzido de caracteres (só letras maiúsculas e alguns sinais de pontuação) e da compactação dos códigos resultantes em um número menor de bits. Examinamos também o uso da estatística de ocorrência de letras e outros símbolos em um texto, para obter um código progressivo de quatro bits. O procedimento garante uma grande eficiência de compressão (cerca de 45%), envolvendo um programa relativamente simples em BASIC, para codificação e decodificação.

Neste artigo, apresentaremos outros métodos de compressão de textos. Um deles, conhecido como *método chinês*, mostra-se particularmente útil para a compressão de textos muito longos e com repetições frequentes de um conjunto reduzido de palavras — o que, certamente, não é próprio de programas de aventuras. De qualquer maneira, esse método completa o leque de alternativas de compressão de textos e, dada a sua eficácia, poderá ser aproveitado em outros tipos de programa.

UM SUPERCOMPRESSOR

O algoritmo estatístico de compressão de textos, que estudamos no artigo anterior, toma como base a diferença de frequência das letras em um texto para construir um sistema de códigos em que os caracteres de maior uso têm um número menor de bits. Para facilitar a programação em BASIC desse algoritmo, limitamos a quatro o número mínimo de bits por caractere. Poderíamos, entretanto, utilizar um número ainda menor de bits para as letras mais frequentes: dois ou três, por exemplo. Com o esquema adotado anteriormente — em que se usa um nibble 0 para assinalar a mudança de “dicionários” de códigos —, os ganhos da compactação seriam mínimos, pois, com a mudança constante de dicionários, o número de zeros acabaria se tornando muito elevado. Portanto, para a codificação com um número menor de bits, aquele esquema não serviria.



■	UM SUPERCOMPRESSOR
■	ESQUEMA DUPLO DE CODIFICAÇÃO
■	CÁLCULO DA FREQUÊNCIA DE PARES

■	DESCOMPRESSÃO
■	O MÉTODO CHINÊS
■	CODIFICAÇÃO POR DICIONÁRIOS
■	DECODIFICAÇÃO
■	VANTAGENS E DESVANTAGENS

ESQUEMA DUPLO DE CODIFICAÇÃO

É possível empregar, porém, um método mais elaborado, que adota um esquema duplo de codificação:

- quanto mais freqüente a letra no texto a ser codificado, menor é o número de bits usado para seu código. Por exemplo: o espaço (caractere mais freqüente em qualquer texto) tem um código binário de três bits (011); a letra E, de cinco bits (10111). Já a letra P, não tão freqüente, tem um código de seis bits (110110), e a letra Z, de oito bits (11111100);

- certos *pares* de letras muito comuns em um texto também recebem um código binário reduzido. O esquema de codificação é o mesmo das letras, mas o efeito de compressão é bem mais poderoso, já que estamos substituindo dois bytes (dezesseis bits) por um número muito menor de bits no novo código.

CÁLCULO DA FREQUÊNCIA DE PARES

Apresentamos, a seguir, um programa destinado a computar e exibir os pares de letras mais freqüentes em um texto. Ele usa a mesma técnica do programa que determina a freqüência simples dos caracteres, fornecido no artigo anterior.

Em primeiro lugar, o programa inicializa os conjuntos **L** — que conterà a freqüência de pares que começa com cada caractere ASCII, de 32 a 90 — e **F** — que conterà em cada casela **F(I,J)** a freqüência acumulada pelo par de caracteres com códigos **I** e **J**.

Em todas as versões, exceto a do Spectrum, o sinal % depois de **L** e **F** serve para definir como inteiro o tipo do conjunto. Isso economiza memória e aumenta a velocidade de processamento. As linhas 20 a 40 não são necessárias nos computadores que inicializam em 0 todas as variáveis numéricas, quando **RUN** é digitado.



```
10 DIM L%(60),F%(60,60),C%(60)
```

```
20 NT=0:NP=0
30 FOR I=1 TO 60:L%(I)=0
35 FOR J=1 TO 60
40 F%(I,J)=0:NEXT J:NEXT I
```

Coloque um comando **CLEAR 3000** na linha 10, antes da declaração **DIM**, para o TRS-80 e TRS-Color.



```
10 DIM L(60),f(60,60),c(60)
20 LET nt=0:LET np=0
30 FOR i=1 TO 60:LET L(i)=0
35 FOR j=1 TO 60
40 LET f(i,j)=0:NEXT j:NEXT i
```

A parte seguinte do programa principal lê as linhas **DATA** que contêm o texto a ser codificado. Para fins de teste, você poderá recorrer ao texto de instruções de uma aventura, dado no artigo anterior, acrescentando-o ao final deste programa.



```
60 PRINT "ANALISANDO..."
70 READ LS:IF LS="*" THEN 100
75 PRINT LS:LT=LEN(LS)
80 NT=NT+LT
85 IF INT(LT/2)<>LT/2 THEN LS=LS+" "
90 GOSUB 520:GOTO 70
```



```
60 PRINT "ANALISANDO..."
70 READ LS:IF LS="*" THEN GOTO 100
75 PRINT LS:LET LT=LEN LS
80 LET nt=nt+lt
85 IF INT LT/2 <> LT/2 THEN LET LS=LS+" "
90 GOSUB 520:GOTO 70
```

Um asterisco na linha de texto (**LS**) indica o fim do mesmo. Caso o texto não tenha terminado, calcula-se o comprimento **LT** de **LS**. Se **LT** for ímpar, é preciso acrescentar um espaço em branco ao final de **LS**, para que a rotina de contagem efetue corretamente o cálculo e a extração de pares de caracteres. Isso é feito pela linha 85, que verifica se o resto da divisão de **LT** por 2

é maior que 0 (se for, é ímpar). Finalmente, a linha 90 chama a rotina de contagem, que começa em 520:



```
500 REM - ROTINA DE CONTAGEM
520 FOR I=1 TO LEN(L$)-1
525 NP=NP+1
530 C1=ASC(MID$(L$,I,1))-31
540 C2=ASC(MID$(L$,I+1,1))-31
545 L$(C1)=L$(C1)+1
550 F$(C1,C2)=F$(C1,C2)+1
560 NEXT I
570 RETURN
```



```
500 REM - ROTINA DE CONTAGEM
520 FOR I=1 TO LEN L$-1
525 LET NP=NP+1
530 LET C1=ASC(L$,I TO I)-31
540 LET C2=ASC(L$,I TO I+1)-31
545 LET L(C1)=L(C1)+1
550 LET F(C1,C2)=F(C1,C2)+1
560 NEXT I
570 RETURN
```

A rotina de contagem é muito simples: o laço que vai da linha 520 à 560 percorre **L\$** tomando um caractere por vez. As variáveis **C1** e **C2** conterão os códigos ASCII do primeiro e do segundo caracteres de um par. Subtraindo o valor 31 desse código, obteremos um número compreendido entre 1 (espaço) e 58 (letra Z). **C1** e **C2** servirão assim como índices para acumular a casela correta de **F** e de **L**.

O programa principal termina quando se atinge o final do texto e a rotina dos resultados é chamada.



```
100 PRINT "TOTAL: ";NT;"CARACTE
RES E ";NP;"PARES"
120 GOSUB 780
140 STOP
```

A rotina de exibição, que começa em 780, utiliza uma rotina de ordenação para mostrar os resultados em ordem decrescente, ou seja, os pares mais frequentes em primeiro lugar.



```
680 REM - ROTINA DE ORDENACAO
690 N=60
695 FOR J=1 TO 60:C$(J)=J:NEXT
700 FL=0
710 N=N-1:FOR J=1 TO N
720 IF F$(I,C$(J))>F$(I,C$(J+1
)) THEN GOTO 740
```

```
730 X=C$(J):C$(J)=C$(J+1):C$(
+1)=X:FL=1
740 NEXT J
750 IF FL=0 OR N=2 THEN RETURN
760 GOTO 700
770 REM - ROTINA DE IMPRESSAO
780 NL=0:PRINT
790 PRINT "FREQUENCIA DE PARES
NO TEXTO":PRINT
800 FOR I=1 TO 60:IF L$(I)=0 TH
EN 850
805 GOSUB 690
810 FOR J=1 TO 60
815 X=C$(J):IF F$(I,X)=0 THEN
GOTO 825
820 PRINT CHR$(I+31)+CHR$(X+31)
:F$(I,X);" ";
825 NEXT J:PRINT
830 NL=NL+1:IF NL<15 THEN GOTO
850
840 NL=0:INPUT "PRESSIONE <ENTE
R> ";X$
850 NEXT I:PRINT
860 RETURN
```



```
680 REM - ROTINA DE ORDENACAO
690 LET N=60
695 FOR J=1 TO 60:LET C(J)=J
700 NEXT J:LET FL=0
710 LET N=N-1:FOR J=1 TO N
720 IF F(I,C(J))>F(I,C(J+1))
THEN GOTO 740
730 LET X=C(J):LET C(J)=C(J+1):
LET C(J+1)=X:FL=1
740 NEXT J
750 IF FL=0 OR N=2 THEN RETURN
760 GOTO 700
770 REM - ROTINA DE IMPRESSAO
780 LET NL=0:PRINT
790 PRINT "FREQUENCIA DE PARES
NO TEXTO":PRINT
800 FOR I=1 TO 60:IF L(I)=0 THE
N GOTO 850
805 GOSUB 690
810 FOR J=1 TO 60
815 LET X=C(J):IF F(I,X)=0 THEN
GOTO 825
820 PRINT CHR$ I+31 +CHR$ X+31
:F(I,X);" ";
825 NEXT J:PRINT
830 LET NL=NL+1:IF NL<15 THEN G
OTO 850
840 LET NL=0:INPUT "PRESSIONE <
ENTER> ";X$
850 NEXT I:PRINT
860 RETURN
```

Para não alterar a matriz **F**, a rotina de ordenação utiliza um conjunto **C**, que funciona como índice — para isso, ele é inicializado no começo da rotina (linha 695), de modo a conter os números 1, 2, 3 etc. A ordenação (do tipo bolha) coloca esses números em uma ordem diferente.

A sub-rotina de impressão examina cada uma das linhas da matriz **F**. Se o total **L(I)** da linha **I** for 0, não se registrou a ocorrência de nenhum caractere

com código **I+31** no texto. Do contrário, o programa chama a rotina de ordenação. O laço que vai da linha 810 à 825 imprime todos os pares cuja frequência é maior que 0. A variável-índice **X**, extraída de **C**, é usada para mostrá-los em ordem decrescente.

Se você quiser ver apenas os dois pares mais frequentes para cada caractere, modifique a linha 810 para:

```
810 FOR J=1 TO 2
```

O MÉTODO CHINÊS

A técnica de compressão explicada a seguir apresenta um alto nível de eficiência, dependendo do texto a codificar e de sua extensão. Trata-se, na realidade, de uma generalização do algoritmo de codificação por pares de letras: se podemos empregar códigos numéricos para representar os pares de caracteres mais frequentes em um texto, a fim de comprimi-lo, é possível também usar triades dos caracteres mais comuns. Naturalmente, essa abordagem se tornaria cada vez mais inviável, à medida que aumentássemos as dimensões da matriz





F: de 3.600 elementos, para contagem de pares, para 216.000 elementos, para contagem de tríades etc.

E por que não montar um dicionário com palavras completas? Cada palavra receberia um código numérico de oito ou de dezesseis bits, e o texto seria composto por uma sequência desses códigos. Para reconstituí-lo, bastaria buscar no dicionário a palavra correspondente a cada número.

Esse sistema é chamado método chinês, pois as palavras em chinês não são formadas a partir de um pequeno conjunto de fonemas ou sílabas, como nos idiomas ocidentais, mas de figuras únicas, os *ideogramas*, que correspondem a códigos.

A idéia pode ser muito boa para uma linguagem natural, mas não devemos nos esquecer de que, no computador, o dicionário também precisa ser armazenado. Assim, para se conseguir o efeito de compressão, é necessário que o resultado da soma do número de bytes obtido no processo com o número de bytes gasto com a armazenagem do dicionário na memória seja menor do que o número de bytes do texto não comprimido (original). A taxa ou eficiência de

compressão é dada pela divisão de um valor pelo outro.

Para entender como funciona esse algoritmo e verificar se é capaz de comprimir o texto-exemplo usado antes, digite e execute este programa.



```
10 DIM F(200),C(200),PS(200),FS
  (50)
20 NT=0:NP=0:NR=0:L
  ET NL=0:NF=0
30 FOR I=1 TO 200
40 F(I)=0:C(I)=I
50 NEXT I
60 PRINT "ANALISANDO..."
70 READ LS:IF LS="*" THEN GOTO
  100
75 NL=NL+1
80 NT=NT+LEN(LS)
85 GOSUB 510
90 NF=NF+LEN(FS(NL)):GOTO 70
100 PRINT "TOTAL: ";NT;"CARACTE
  RES E ";NP;"PALAVRAS"
110 PRINT "DICCIONARIO COM ";NR;
  "CARACTERES."
115 PRINT "TEXT0 COMPRIMIDO COM
  ";NF;"CARACTERES"
120 PRINT "ORDENANDO..."
130 GOSUB 690:GOSUB 780
```

```
250 STOP
500 REM - ROTINA DE CONTAGEM
510 LS=LS+" ":I=1:L=1
520 C=ASC(MID$(LS,I,1))
525 IF C<>32 THEN GOTO 560
540 I=I+1:IF I<LEN(LS) THEN
  GOTO 520
550 RETURN
560 L=I
570 C=ASC(MID$(LS,I,1))
575 IF C=32 THEN GOTO 610
590 I=I+1:IF I<LEN(LS) THEN
  GOTO 570
600 RETURN
610 XS=MID$(LS,L,I-L)
620 FOR N=1 TO NP
630 IF XS<>PS(N) THEN GOTO 640
635 F(N)=F(N)+1:FS(NL)=FS(NL)+
  CHR$(N):GOTO 520
640 NEXT N
650 NP=NP+1:F(NP)=1:PS(NP)=XS
660 NR=NR+LEN(XS):FS(NL)=FS(NL)+
  +CHR$(NP)
670 GOTO 520
680 REM - ROTINA DE ORDENACAO
690 NR=NP
700 FL=0
710 N=N-1:FOR I=1 TO N
720 IF F(C(I))>F(C(I+1)) THEN
  GOTO 740
730 X=C(I):C(I)=C(I+1)
735 C(I+1)=X:FL=1
740 NEXT I
```



```

750 IF FL=0 OR N=2 THEN RETURN
760 GOTO 700
770 REM - ROTINA DE IMPRESSAO
780 NX=0
790 PRINT "FREQUENCIA DE PALAVR
AS NO TEXTO":PRINT
800 FOR I=1 TO NP
820 PRINT I;PS(C(I)),F(C(I))
830 NX=NX+1:IF NX<15 THEN GOTO
850
840 NX=0:INPUT "PRESSIONE <ENTE
R> ";XS
850 NEXT I:PRINT
860 RETURN

```

Os usuários do TRS-80 e do TRS-Color devem colocar um comando **CLEAR 3000** antes do **DIM**, na linha 10.

S

```

10 DIM f(200),c(200),p$(200,15)
,f$(50)
20 LET nt=0:LET np=0:LET nt=0:L
ET nl=0:LET nf=0
30 FOR i=1 TO 200
40 LET f(i)=0:LET c(i)=i
50 NEXT i
60 PRINT "ANALISANDO..."
70 READ LS:IF LS="*" THEN GOTO
100
75 LET nl=nl+1
80 LET nt=nt+LEN LS
85 GOSUB 510
90 LET nf=nf+LEN f$(nl):GOTO 70
100 PRINT "TOTAL: ";nt;"CARACTE
RES ";np;"PALAVRAS"
110 PRINT "DICCIONARIO COM ";nr;
"CARACTERES."
115 PRINT "TEXTO COMPRIMIDO COM
";nf;"CARACTERES"
120 PRINT "ORDENANDO..."
130 GOSUB 690:GOSUB 780
250 STOP

```

```

500 REM - ROTINA DE CONTAGEM
510 LET LS=LS+" ";LET i=1:LET L
=1
520 LET c=ASC LS(i TO i)
525 IF c<>32 THEN GOTO 560
540 LET i=i+1:IF i<LEN LS THEN
GOTO 520
550 RETURN
560 LET L=i
570 LET c=ASC LS(i TO i)
575 IF c=32 THEN GOTO 610
590 LET i=i+1:IF i<LEN LS THEN
GOTO 570
600 RETURN
610 LET x$=LS(L TO I-1)
620 FOR n=1 TO NP
630 IF x$<>p$(n) THEN GOTO 640
635 LET f(n)=f(n)+1:LET f$(n1)=
f$(n1)+CHRS(n):GOTO 520
640 NEXT n
650 LET np=np+1:LET f(np)=1:LET
p$(np)=x$
660 LET nr=nr+LEN x$:LET f$(n1
)=f$(n1)+CHRS(np)
670 GOTO 520
680 REM - ROTINA DE ORDENACAO

```

```

690 LET n=np
700 LET f1=0
710 LET n=n-1:FOR i=1 TO n
720 IF f(c(i))>f(c(i+1)) THEN
GOTO 740
730 LET x=c(i):LET c(i)=c(i+1):
LET c(i+1)=x:LET f1=1
740 NEXT i
750 IF f1=0 OR n=2 THEN RETURN
760 GOTO 700
770 REM - ROTINA DE IMPRESSAO
780 LET nx=0
790 PRINT "FREQUENCIA DE PALAVR
AS NO TEXTO":PRINT
800 FOR i=1 TO np
820 PRINT i;p$(c(i)),f(c(i))
830 LET nx=nx+1:IF nx<15 THEN G
OTO 850
840 LET nx=0:PRINT "PRESSIONE <
ENTER> ";INPUT x$
850 NEXT i:PRINT
860 RETURN

```

A parte principal do programa é a rotina de extração de palavras, que vai da linha 500 à 670.

Para identificar e separar as palavras contidas em uma linha de texto, essa rotina procura o primeiro caractere não-branco (que não é um espaço) a partir do caractere 1 da linha **LS**. A posição inicial da palavra é armazenada na variável **L** (linha 560).

Em seguida, a rotina procura o fim da palavra: continua a percorrer o texto até encontrar o primeiro caractere de espaço. Isso é feito pelas linhas 570 a 600. Note que, para evitar que a última palavra de um texto se perca, a linha 510 da rotina sempre acrescenta um espaço em branco após **LS**.

Achado o final da palavra, o *sub-string* compreendido entre **L** e **I-1** (a nova posição) é copiado em **X\$** (linha 610). As linhas 620 a 660 verificam então se a palavra **X\$** existe no dicionário **PS**, com **NP** palavras. Se não existe, a nova palavra é acrescentada. Antes de voltar para a linha 520, para buscar nova palavra no texto, a rotina incrementa em **F** a frequência de ocorrência da palavra. A linha de saída, que contém o texto comprimido (armazenada em **F\$**), recebe um novo caractere, que corresponde ao código da palavra no dicionário. Portanto, uma palavra inteira é substituída por um byte.

Concluindo o programa, as rotinas das linhas 690 e 780 colocam as palavras do dicionário em ordem alfabética, exibindo-as na tela em grupos de quinze, como mostramos a seguir:

FREQUENCIA DAS PALAVRAS	
DE	15
O	10
A	8
E	8

UM	6
DO	5
SUA	5
...	...

Observamos no texto-exemplo que:

- as palavras curtas aparecem mais;
- há um grande número de palavras com uma única ocorrência;
- vírgulas e outros sinais são tratados como parte da palavra junto à qual se encontram.

Com isso, a eficiência de compressão do algoritmo é muito baixa: de 1228 caracteres do texto original, conseguimos comprimir apenas 215 caracteres (uma taxa de 82,5%); porém, o dicionário ocupa outros 870 bytes, dando um total de 1085. A eficiência se reduz, assim, a apenas 11,6%.

O algoritmo é mais eficaz quando se combinam textos bem longos e uso de um vocabulário reduzido.

DECODIFICAÇÃO

Para ver como o texto comprimido é reconstituído, acrescente a sub-rotina de decodificação:

S

```

140 GOSUB 900
890 REM - ROTINA DECODIFICACAO
900 NX=0:FOR I=1 TO NL
910 FOR J=1 TO LEN(F$(I))
920 PRINT " ";PS(ASC(MIDS(F$(I),
J,1)))
930 NEXT J:PRINT
940 NX=NX+1:IF NX<15 THEN GOTO
950
945 NX=0:INPUT "PRESSIONE <ENTE
R> ";RS
950 NEXT I:RETURN

```

S

```

140 GOSUB 900
890 REM - ROTINA DECODIFICACAO
900 LET nx=0:FOR i=1 TO nL
910 FOR j=1 TO LEN f$(i)
920 PRINT " ";p$(ASC(f$(i,j TO
j)))
930 NEXT j:PRINT
940 LET nx=nx+1:IF nx<15 THEN G
OTO 950
945 LET nx=0:PRINT "PRESSIONE <

```

Essa rotina funciona de modo oposto ao da de codificação, mas é bem mais simples. Os códigos armazenados em **F\$** são recuperados pela função **ASC** da linha 920, e servem como índice do dicionário **PS** para reconstruir o texto original. Os espaços entre palavras são inseridos automaticamente.

ROTINAS EM CÓDIGO DE MÁQUINA (2)

■	ROTINAS NAS LINHAS DATA
■	A ESTRUTURA DE UMA LINHA
■	PEEK E POKE
■	ENTENDA COMO FUNCIONA

A linha **DATA** é um excelente lugar para se colocar programas em linguagem de máquina embutidos em um programa em BASIC. Aprenda aqui os truques que facilitam essa tarefa no MSX.



Com freqüência, armazenamos, em algum lugar da memória, uma rotina em código de máquina, um arquivo de padrões de um desenho ou até mesmo uma composição musical acompanhada de instruções. Muitas vezes é útil incorporar esse arquivo de rotinas a um programa em linguagem BASIC. Se colocarmos seus códigos nas linhas **DATA** do programa que os acessa, tornaremos bem mais fácil o manuseio do sistema, viabilizando a obtenção de uma listagem completa e evitando repetidas buscas no gravador ou no drive, para uma posterior junção. Com essas linhas **DATA**, precisaremos apenas acrescentar um laço que, por intermédio do comando **POKE**, carregue uma determinada região da memória com os dados nela contidos.

INTRODUÇÃO DAS LINHAS DATA

Para apresentar os códigos do arquivo, o mais conveniente é utilizar números hexadecimais, que ocupam menos espaço e reduzem a possibilidade de se cometer erros na digitação. Além disso, o modo como dispomos os números na linha pode facilitar correções ou consultas posteriores — ou seja, a quantidade de números por linha indica que ali se encontra um determinado padrão gráfico ou que aqueles códigos representam uma pequena sub-rotina dentro da rotina principal.

O programa a seguir irá auxiliá-lo nessa tarefa. Inicialmente, você irá complementá-lo com linhas **DATA** seguidas de caracteres que depois serão superpostos pelo arquivo. É importante ressaltar que a quantidade de caracteres por linha determinará o número de código nela armazenados. Assim, você pode

“moldar” as linhas **DATA**, dando-lhes a aparência que julgar melhor. Por exemplo, uma linha do tipo:

```
330 DATA XXXXXXXX
```

seria transformada em:

```
330 DATA A3,F2,01
```

Convém, portanto, que você gaste algum tempo na criação dessas linhas, se quiser uma listagem bem organizada.

RODANDO O PROGRAMA

Após o acréscimo das linhas **DATA**, passamos à execução do programa. Em primeiro lugar, ele solicita o endereço inicial do arquivo, pedindo, em seguida, seu comprimento. Depois de termos digitado essas informações, o computador pergunta se as linhas **DATA** já foram introduzidas. Caso isso ainda não tenha sido feito, o programa será interrompido. Finalmente, ele pede o número da primeira linha a ser preenchida. Se não encontrar essa linha, o computador interromperá a execução do programa e imprimirá uma mensagem de erro. O mesmo irá ocorrer se a linha indicada não contiver o comando **DATA**.

É importante que todas as linhas reservadas para os códigos estejam em seqüência na listagem — durante a execução, uma mensagem de erro pára o programa quando o computador encontra uma linha que não seja **DATA**.

```
10 CLS
20 INPUT "Qual o endereço do arquivo ";EN:PRINT
30 INPUT "Qual o comprimento ";C:PRINT
40 PRINT "Você já introduziu as linhas DATA? S/N ":PRINT
50 GS=INKEYS:IF GS="" THEN GOTO 50
60 IF GS="N" OR GS="n" THEN PRINT "Eu vou parar o programa para você fazer isso":STOP
70 INPUT "Qual a primeira linha DATA ";LD:PRINT
80 X=32769!
90 IF PEEK(X)+256*PEEK(X+3)=LD THEN 130
100 IF PEEK(X)+256*PEEK(X+1)=0 THEN PRINT "Não achei esta linha ":STOP
```

```
110 X=PEEK(X)+256*PEEK(X+1)
120 GOTO 90
130 IF PEEK(X+4)<>132 THEN PRINT "Esta não é uma linha DATA":PRINT:GOTO 70
140 X=X+3
150 FOR I=0 TO C-1
160 X=X+3
170 IF PEEK(X)=0 THEN X=X+5:GOSUB 310
180 IF PEEK(X+1)=0 THEN POKE X,32:X=X+6:GOSUB 310
190 LET AS=HEX$(PEEK(X+1))
200 IF LEN(AS)=1 THEN AS="0"+AS
210 POKE X,ASC(MID$(AS,1,1))
220 POKE X+1,ASC(MID$(AS,2,1))
230 IF PEEK(X+2)=0 THEN X=X+1:GOTO 270
240 IF PEEK(X+3)=0 THEN POKE (X+2),32:GOTO 270
250 IF PEEK(X+4)=0 THEN POKE (X+2),32:POKE (X+3),32:X=X+1:GOTO 270
260 IF I>C-1 THEN POKE(X+2),4
270 NEXT I
280 IF PEEK(X+2)<>0 THEN POKE(X+2),32:X=X+1:GOTO 280
290 LIST
300 STOP
310 IF PEEK(X)<>132 THEN PRINT "Faltam linhas DATA":STOP ELSE X=X+2
320 RETURN
```

LEITURA DA MEMÓRIA

Para entender o funcionamento do programa, é necessário conhecer a estrutura de uma linha **DATA**, ou seja, saber como ela é armazenada na RAM.

Ao ligar o computador, digite:

```
10 DATA A,B,C
```

Agora, para “ler” o que está escrito na memória, usaremos o comando **PEEK**, que mostra o conteúdo de um determinado endereço. Começaremos pelo exame do byte de número 32769, que contém a primeira informação da linha inicial de um programa em BASIC. Para isso, digite o comando direto:

```
FOR I=32769 TO 32782:PRINT PEEK(I); " ";:NEXT I
```

Você deve ter obtido a seguinte seqüência de números:

```
13 128 10 0 132 32 65 44 66 44
67 0 0 0
```


ESTRUTURA DA LINHA

Os dois primeiros números referem-se ao endereço inicial da próxima linha. Esse endereço é armazenado na forma **LH** (do inglês **Low**, parte baixa, e **High**, parte alta). Pode ser decodificado do seguinte modo: $128 \times 256 + 13 = 32781$. O par seguinte equivale ao número da linha, que é armazenado da mesma forma: $0 \times 256 + 10 = 10$. O próximo número, 132, é o código da palavra **DATA**.

O MSX atribui um número — ou *token* (símbolo, indicação) — a cada palavra e caractere reservados. Assim, o computador economiza bastante memória: em vez de armazenar todos os caracteres da palavra, guarda apenas o seu *token*. É importante não confundir os *tokens* com os códigos ASCII, que representam os caracteres comuns.

O número 32, que vem a seguir, é o código ASCII do espaço em branco, que separa os dados do comando **DATA**. Os próximos cinco bytes contêm esses dados, que vêm separados por uma vírgula (44). O primeiro 0, na seqüência, indica o fim de uma linha, e os dois outros, o fim do programa. Quando todas as linhas tiverem sido examinadas, os dois bytes iniciais apontam para o primeiro destes dois zeros.

FUNCIONAMENTO DO PROGRAMA

Após conseguir as informações solicitadas no início do programa, o computador passa às verificações.

A linha 80 inicializa a variável principal, **X**, que contém o endereço do byte em estudo. Primeiro, ela assume o valor 32769, que, conforme vimos, é o endereço do byte inicial da primeira linha. A linha 90 verifica o terceiro e quarto bytes, obtendo o número da linha. Se esta for a linha procurada (a primeira linha **DATA**), o programa é desviado para a linha 130.

A linha 100 pesquisa o primeiro e o segundo bytes, que contêm o endereço da próxima linha. Se houver um número 0 no lugar do endereço, estamos no fim do programa — o que significa que a linha não foi encontrada. O computador imprime então uma mensagem de erro. Caso contrário, é necessário pesquisar a próxima linha. Assim, depois de atribuir à variável **X** o valor encontrado no primeiro e no segundo bytes (linha 110), o programa retorna para a linha 90. Esse laço só será interrompido quando a próxima linha do programa foi localizada ou quando chegarmos ao final da listagem.

Ao encontrar a linha indicada pelo usuário, o microcomputador verifica se se trata de uma linha **DATA**. Para isso, pesquisa seu quinto byte. Se ele não contiver o *token* correspondente à instrução **DATA**, será impressa uma mensagem de erro.

O LAÇO PRINCIPAL

Feitas todas essas verificações, o programa inicia o laço principal, controlado pela variável **I**, que coloca os códigos de máquina no programa em **BASIC**. Antes, porém, a linha 140 acrescenta três unidades a **X**, para que, dentro do laço, essa variável seja incrementada em mais três e alcance o endereço do primeiro caractere da linha que deverá ser preenchida.

Primeiro, o laço principal busca o código na região da memória determinada pelo usuário, transforma-o na notação hexadecimal (linha 190) e o coloca sobre os caracteres para os quais reservamos espaço nas linhas **DATA** (linhas 210 e 220). Se esse código for inferior a 15, sua notação em hexadecimal terá apenas um caractere. Para dar uma melhor apresentação ao programa, a linha 200 acrescentará um 0 à direita desse caractere.

As linhas 170 e 180 são responsáveis pela mudança de linha. Se o byte em questão contiver o valor 0, um salto será necessário, pois esse valor indica o fim da linha. Incrementa-se então a variável **X**, de modo que ela alcance o endereço do próximo caractere a ser superposto por um código, na linha seguinte. Porém, se o byte posterior a este contiver um 0, não será possível colocar um código na linha, por falta de espaço. Nesse caso, além do incremento, é preciso imprimir um espaço em branco sobre o caractere que se encontra no endereço anterior ao 0. Ambas as linhas, antes do salto, utilizam uma sub-rotina (linha 310) para verificar se a próxima linha é realmente uma linha **DATA**.

Convém estar atento para que nada seja impresso após o último código, pois ele deve finalizar a linha **DATA**. Assim, quando o código já foi introduzido nas posições **X** e **X+1**, não haverá vírgula se o primeiro, o segundo ou o terceiro bytes após ele contiverem o número 0. Nesse caso, alguns espaços em branco serão colocados sobre os caracteres anteriores àqueles bytes. O valor de **X** também sofrerá uma pequena mudança para que, na próxima volta do laço, o endereço **X**, incrementado em três, corresponda ao endereço de 0, e a linha 170

se encarregue de promover o salto para a próxima linha **DATA**.

Se o primeiro, o segundo e o terceiro bytes após o código não contiverem 0, a linha 260 se encarregará de imprimir uma vírgula.

Quando o programa sai do laço, a linha 280 limpa a última linha **DATA**, imprimindo espaços em branco sobre os caracteres que ali se encontravam.

Finalmente, o computador exhibe na tela a listagem completa, mostrando ao usuário os códigos já incorporados ao programa.

Como estamos fazendo modificações dentro do programa, convém gravá-lo antes da execução. Um pequeno erro de digitação pode provocar um **POKE** e inutilizar o programa.

LISTA DE TOKENS

Com os novos conhecimentos sobre a armazenagem das linhas em **BASIC**, você pode fazer uma série de experiências com o comando **POKE**. Tente, por exemplo, digitar uma linha e mudar o seu número com um **POKE** no primeiro e no segundo bytes. Lembre-se de que o primeiro byte do **BASIC** é o de número 32769. Você pode também “olhar” as linhas armazenadas no **BASIC** recorrendo a alguns comandos diretos, como fizemos em nosso exemplo.

Provavelmente, será útil dispor de uma lista dos comandos com seus respectivos *tokens*. Para obtê-la, digite e execute este programa:

```
10 E=14962
20 C=65
30 PRINT CHR$(C);
40 A=PEEK(E)
50 B=PEEK(E+1)
60 AS=CHR$(A)
70 IF A<128 THEN PRINT AS;:GOTO 110
80 PRINT CHR$(A-128);TAB(8);B
90 E=E+1
100 IF PEEK(E+1)<>0 THEN PRINT CHR$(C);
110 IF PEEK(E+1)<>0 THEN 150
120 C=C+1:IF C=89 THEN 170
125 BS=CHR$(C):PRINT
130 IF BS="J" OR BS="Q" THEN 150
140 PRINT BS;
150 E=E+1
160 IF E<=15649 THEN 40
170 E=15654:PRINT
180 A=PEEK(E):B=PEEK(E+1)
190 IF A>127 THEN PRINT CHR$(A-128);
200 IF A<128 THEN PRINT CHR$(A);
210 PRINT TAB(8);B
220 E=E+2:PRINT
230 IF E<15673 THEN GOTO 180
```


LINHA	FABRICANTE	MODELO	FABRICANTE	MODELO	PAÍS	LINHA
Apple II +	Appletronica	Thor 2010	Appletronica	Thor 2010	Brasil	Apple II +
Apple II +	CCE	MC-4000 Exato	Apply	Apply 300	Brasil	Sinclair ZX-81
Apple II +	CPA	Absolutus	CCE	MC-4000 Exato	Brasil	Apple II +
Apple II +	CPA	Polaris	CPA	Absolutus	Brasil	Apple II +
Apple II +	Digitus	DGT-AP	CPA	Polaris	Brasil	Apple II +
Apple II +	Dismac	D-8100	Codimex	CS-6508	Brasil	TRS-Color
Apple II +	ENIAC	ENIAC II	Digitus	DGT-100	Brasil	TRS-80 Mod.III
Apple II +	Franklin	Franklin	Digitus	DGT-1000	Brasil	TRS-80 Mod.III
Apple II +	Houston	Houston AP	Digitus	DGT-AP	Brasil	Apple II +
Apple II +	Magnex	DM II	Dismac	D-8000	Brasil	TRS-80 Mod. I
Apple II +	Maxitronica	MX-2001	Dismac	D-8001/2	Brasil	TRS-80 Mod. I
Apple II +	Maxitronica	MX-48	Dismac	D-8100	Brasil	Apple II +
Apple II +	Maxitronica	MX-64	Dynacom	MX-1600	Brasil	TRS-Color
Apple II +	Maxitronica	Maxitronic I	ENIAC	ENIAC II	Brasil	Apple II +
Apple II +	Microcraft	Craft II Plus	Engebras	AS-1000	Brasil	Sinclair ZX-81
Apple II +	Milmar	Apple II Plus	Filcres	NEZ-8000	Brasil	Sinclair ZX-81
Apple II +	Milmar	Apple Master	Franklin	Franklin	USA	Apple II +
Apple II +	Milmar	Apple Senior	Gradiente	Expert GPC1	Brasil	MSX
Apple II +	Omega	MC-400	Houston	Houston AP	Brasil	Apple II +
Apple II +	Polymax	Maxxl	Kemtron	Naja 800	Brasil	TRS-80 Mod.III
Apple II +	Polymax	Poly Plus	LNW	LNW-80	USA	TRS-80 Mod. I
Apple II +	Spectrum	Microengenho I	LZ	Color 64	Brasil	TRS-Color
Apple II +	Spectrum	Spectrum ed	Magnex	DM II	Brasil	Apple II +
Apple II +	Suporte	Venus II	Maxitronica	MX-2001	Brasil	Apple II +
Apple II +	Sycomig	SIC I	Maxitronica	MX-48	Brasil	Apple II +
Apple II +	Unitron	AP II	Maxitronica	MX-64	Brasil	Apple II +
Apple II +	Victor do Brasil	Elppa II Plus	Maxitronica	Maxitronic I	Brasil	Apple II +
Apple II +	Victor do Brasil	Elppa Jr.	Microcraft	Craft II Plus	Brasil	Apple II +
Apple IIe	Microcraft	Craft IIe	Microcraft	Craft IIe	Brasil	Apple IIe
Apple IIe	Microdigital	TK-3000 IIe	Microdigital	TK-3000 IIe	Brasil	Apple IIe
Apple IIe	Spectrum	Microengenho II	Microdigital	TK-82C	Brasil	Sinclair ZX-81
MSX	Gradiente	Expert GPC-1	Microdigital	TK-83	Brasil	Sinclair ZX-81
MSX	Sharp	Hotbit HB-8000	Microdigital	TK-85	Brasil	Sinclair ZX-81
Sinclair Spectrum	Microdigital	TK-90X	Microdigital	TK-90X	Brasil	Sinclair Spectrum
Sinclair Spectrum	Timex	Timex 2000	Microdigital	TKS-800	Brasil	TRS-Color
Sinclair ZX-81	Apply	Apply 300	Milmar	Apple II Plus	Brasil	Apple II +
Sinclair ZX-81	Engebras	AS-1000	Milmar	Apple Master	Brasil	Apple II +
Sinclair ZX-81	Filcres	NEZ-8000	Milmar	Apple Senior	Brasil	Apple II +
Sinclair ZX-81	Microdigital	TK-82C	Multix	MX-Compacto	Brasil	TRS-80 Mod.IV
Sinclair ZX-81	Microdigital	TK-83	Omega	MC-400	Brasil	Apple II +
Sinclair ZX-81	Microdigital	TK-85	Polymax	Maxxl	Brasil	Apple II +
Sinclair ZX-81	Prologica	CP-200	Polymax	Poly Plus	Brasil	Apple II +
Sinclair ZX-81	Ritas	Ringo R-470	Prologica	CP-200	Brasil	Sinclair ZX-81
Sinclair ZX-81	Timex	Timex 1000	Prologica	CP-300	Brasil	TRS-80 Mod.III
Sinclair ZX-81	Timex	Timex 1500	Prologica	CP-400	Brasil	TRS-Color
TRS-80 Mod. I	Dismac	D-8000	Prologica	CP-500	Brasil	TRS-80 Mod.III
TRS-80 Mod. I	Dismac	D-8001/2	Ritas	Ringo R-470	Brasil	Sinclair ZX-81
TRS-80 Mod. I	LNW	LNW-80	Sharp	Hotbit HB-8000	Brasil	MSX
TRS-80 Mod. I	Video Genie	Video Genie I	Spectrum	Microengenho I	Brasil	Apple II +
TRS-80 Mod.III	Digitus	DGT-100	Spectrum	Microengenho II	Brasil	Apple IIe
TRS-80 Mod.III	Digitus	DGT-1000	Spectrum	Spectrum ed	Brasil	Apple II +
TRS-80 Mod.III	Kemtron	Naja 800	Suporte	Venus II	Brasil	Apple II +
TRS-80 Mod.III	Prologica	CP-300	Sycomig	SIC I	Brasil	Apple II +
TRS-80 Mod.III	Prologica	CP-500	Sysdata	Sysdata III	Brasil	TRS-80 Mod.III
TRS-80 Mod.III	Sysdata	Sysdata III	Sysdata	Sysdata IV	Brasil	TRS-80 Mod.IV
TRS-80 Mod.III	Sysdata	Sysdata Jr.	Sysdata	Sysdata Jr.	Brasil	TRS-80 Mod.III
TRS-80 Mod.IV	Mullix	MX-Compacto	Timex	Timex 1000	USA	Sinclair ZX-81
TRS-80 Mod.IV	Sysdata	Sysdata IV	Timex	Timex 1500	USA	Sinclair ZX-81
TRS-Color	Codimex	CS-6508	Timex	Timex 2000	USA	Sinclair Spectrum
TRS-Color	Dynacom	MX-1600	Unitron	AP II	Brasil	Apple II +
TRS-Color	LZ	Color 64	Victor do Brasil	Elppa II Plus	Brasil	Apple II +
TRS-Color	Microdigital	TKS-800	Victor do Brasil	Elppa Jr.	Brasil	Apple II +
TRS-Color	Prologica	CP-400	Video Genie	Video Genie I	USA	TRS-80 Mod. I

INPUT foi especialmente projetado para microcomputadores compatíveis com as sete principais linhas existentes no mercado.

Os blocos de textos e listagens de programas aplicados apenas a determinadas linhas de micros podem ser identificados por meio dos seguintes símbolos:



Sinclair ZX-81



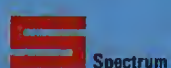
TRS-80



TK-2000



MSX



Spectrum



TRS-Color



Apple II

Quando o emblema for seguido de uma faixa, então tanto o texto como os programas que se seguem passam a ser específicos para a linha indicada.

O Diário de Anne Frank emocionou milhões. E era só uma parte da história.

Anne Frank é lembrada como símbolo do extermínio de milhões de judeus. Seu drama e o de sua família são vistos agora de um ângulo diferente, o da mulher holandesa que ajudou os Frank durante os dois anos que passaram escondidos dos nazistas num sótão de Amsterdã, e que se tornou o único elo entre eles e o mundo exterior. É com tocante simplicidade que esta história triste — o “outro lado” do *Diário* de Anne Frank — é contada neste livro, poderoso testemunho da coragem de que é capaz o ser humano, mesmo nos momentos sombrios da História.



Magnificamente escrito por uma pessoa devotada ao ser humano. Seu estilo simples cativa o leitor. Um livro recomendável aos que se preocupam com os rumos do mundo.”

— Isaac Bashevis Singer, Prêmio Nobel de Literatura de 1978



EDITORA BEST SELLER

JÁ NAS
LIVRARIAS